

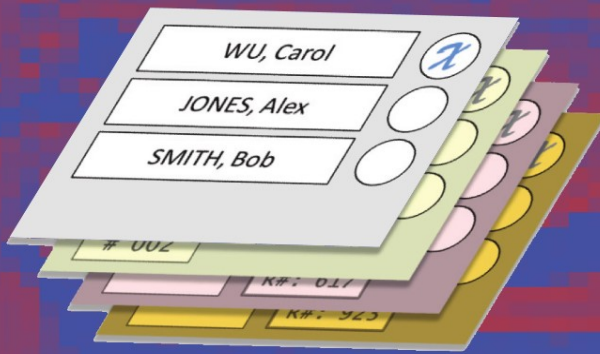
State-of-the-Art
Survey

LNCS 6000

David Chaum Markus Jakobsson
Ronald L. Rivest Peter Y.A. Ryan
Josh Benaloh Miroslaw Kutylowski
Ben Adida (Eds.)

Towards Trustworthy Elections

New Directions in Electronic Voting



 Springer

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Alfred Kobsa

University of California, Irvine, CA, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

TU Dortmund University, Germany

Madhu Sudan

Microsoft Research, Cambridge, MA, USA

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max-Planck Institute of Computer Science, Saarbruecken, Germany

David Chaum Markus Jakobsson
Ronald L. Rivest Peter Y.A. Ryan
Josh Benaloh Miroslaw Kutylowski
Ben Adida (Eds.)

Towards Trustworthy Elections

New Directions in Electronic Voting

Volume Editors

David Chaum

Independent researcher; david@chaum.com

Markus Jakobsson

Palo Alto Research Center, CA, USA; markus.jakobsson@gmail.com

Ronald L. Rivest

Massachusetts Institute of Technology, Cambridge, MA, USA; rivest@mit.edu

Peter Y.A. Ryan

University of Luxembourg, Luxembourg; peter.ryan@uni.lu

Josh Benaloh

Microsoft Research Center, Redmond, WA, USA; benaloh@microsoft.com

Mirosław Kutylowski

Wrocław University of Technology, Poland; mirekk@swarog.im.pwr.wroc.pl

Ben Adida

Harvard Center for Research on Computation and Society Boston, MA, USA
ben@adida.net

Administrative Editor

Richard Carback

University of Maryland Baltimore County, Baltimore, MD, USA
carback1@umbc.edu

Library of Congress Control Number: 2010927806

CR Subject Classification (1998): E.3, K.6.5, D.4.6, C.2, J.1, H.4

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN 0302-9743

ISBN-10 3-642-12979-X Springer Berlin Heidelberg New York

ISBN-13 978-3-642-12979-7 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

springer.com

© IAVOSS/Springer-Verlag Berlin Heidelberg 2010

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper 06/3180

Foreword

Cryptography applied to elections holds the promise of being:

- A critical step in a centuries-long and hard-fought struggle for voting rights.
- An end to disputed elections justifying power grabs and the erosion of democracy.
- A stop to a legacy of profit from bungled technology and outmoded analysis.
- A potential reversal of declining voter confidence.
- A breakthrough that for the first time lets voters verify that their own votes are counted.
- A scalable system for secret ballot elections with transparent integrity.
- A technology capable of taking democracy to new levels.

If you are interested in this challenging problem—a problem that even school children can understand but that is made harder than traditional computer security by requirements for public verifiability and ballot secrecy—then this volume is for you. Just as cryptography can keep messages secure for senders no matter the routing to the destination, in principle it can keep votes secure for voters, from vote casting all the way through to inclusion of votes in the final tally. The challenge addressed here is to find practical means suitable for actual elections.

In 2001, Ron Rivest and myself invited all those we could find who had published more than one academic paper on voting security to a “Workshop On Trustworthy Elections.” Almost all of them attended. This first workshop on the subject took place in a room built nearly a century earlier to house, fittingly, technology for the first public global radio communication system. A remarkable consensus emerged during the discussion session on the last day, the conclusion that cryptography holds real promise to improve elections and a decision to explore it further.

A series of WOTE workshops ensued, sometimes under names adapted to sponsoring organizations, and ultimately resulting in an annual event sponsored by its own international association—the International Association for Voting Systems Sciences. In addition to this open series of conferences, there was also a week-long invitation-only workshop in 2007.

This volume represents, for each of these meetings, papers selected by a key member of the respective Program Committee serving as an invited editor. The volume aims to be comprehensive as far as the flavor and scope of the field and to bring together important but previously unpublished works. It should prove a valuable resource for those curious about, entering, or seeking a deeper understanding of this extraordinarily important and open-ended new field.

Organization

WOTE 2001

Workshop on Trustworthy Elections

August 2001 Marconi Conference Center, Tomales Bay, California

Organizers: Ronald L. Rivest, *David Chaum*

DIMACS Workshop on Electronic Voting—Theory and Practice

May 2004 DIMACS Center, Rutgers University

Organizers: *Markus Jakobsson*, Ari Juels

Workshop Frontiers in Electronic Elections

September 2005 University of Milan

Program Committee: Christian Cachin, Jean-Jacques Quisquater, *Ronald L. Rivest*, Peter Y.A. Ryan, Berry Schoenmakers

WOTE 2006

June 2006 Cambridge University

General Chair: *Peter Y.A. Ryan*

WOTE 2007

June 2007 University of Ottawa

Program/General Chair: *Josh Benaloh*

Frontiers of Electronic Voting

June/July 2007 Schloss Dagstuhl

Organizers: David Chaum, *Mirosław Kutylowski*, Ronald L. Rivest, Peter Y.A. Ryan

WOTE 2008

July 2008 Katholieke Universiteit Leuven, Belgium

Chairs: *Ben Adida*, Olivier Pereira

The names of editors of this volume appear in italics above. Financial and other support for these six events was provided, respectively, in part by:

- (1) CalTech/MIT Voting Technology Project and the organizers
- (2) Center for Discrete Mathematics and Theoretical Computer Science, Rutgers and Princeton Universities
- (3) European Network of Excellence in Cryptology in association with the 10th European Symposium on Research in Computer Security
- (4) IAVOSS in association with the Privacy Enhancing Technologies Symposium 2006
- (5) IAVOSS in association with the Privacy Enhancing Technologies Symposium 2007
- (6) Schloss Dagstuhl, Leibniz Center for Informatics
- (7) Katholieke Universiteit Leuven

Table of Contents

The Witness -Voting System	1
<i>Ed Gerck</i>	
Coercion-Resistant Electronic Elections	37
<i>Ari Juels, Dario Catalano, and Markus Jakobsson</i>	
Receipt-Free K -out-of- L Voting Based on ElGamal Encryption	64
<i>Martin Hirt</i>	
A Secure Architecture for Voting Electronically (SAVE)	83
<i>Jonathan A. Goler and Edwin J. Selker</i>	
A Modular Voting Architecture (“Frog Voting”)	97
<i>Shuki Bruck, David Jefferson, and Ronald L. Rivest</i>	
Unconditionally Secure Electronic Voting	107
<i>Akira Otsuka and Hideki Imai</i>	
Electronic Elections: A Balancing Act	124
<i>Pedro A.D. Rezende</i>	
An Implementation of a Mix-Net Based Network Voting Scheme and Its Use in a Private Organization	141
<i>Jun Furukawa, Kengo Mori, and Kazue Sako</i>	
The Vector-Ballot Approach for Online Voting Procedures	155
<i>Aggelos Kiayias and Moti Yung</i>	
On Optical Mark-Sense Scanning	175
<i>Douglas W. Jones</i>	
On Some Incompatible Properties of Voting Schemes	191
<i>Benoît Chevallier-Mames, Pierre-Alain Fouque, David Pointcheval, Julien Stern, and Jacques Traoré</i>	
A Threat Analysis of Prêt à Voter	200
<i>Peter Y.A. Ryan and Thea Peacock</i>	
Anonymity in Voting Revisited	216
<i>Hugo Jonker and Wolter Pieters</i>	
Anonymous One-Time Broadcast Using Non-interactive Dining Cryptographer Nets with Applications to Voting	231
<i>Jeroen van de Graaf</i>	

An Introduction to PunchScan	242
<i>Stefan Popoveniuc and Ben Hosp</i>	
Component Based Electronic Voting Systems	260
<i>David Lundin</i>	
A Verifiable Voting Protocol Based on Farnel (Extended Abstract)	274
<i>Roberto Araújo, Ricardo Felipe Custódio, and Jeroen van de Graaf</i>	
Verifying Privacy-Type Properties of Electronic Voting Protocols: A Taster	289
<i>Stéphanie Delaune, Steve Kremer, and Mark Ryan</i>	
Improving Remote Voting Security with CodeVoting	310
<i>Rui Joaquim, Carlos Ribeiro, and Paulo Ferreira</i>	
A Practical and Secure Coercion-Resistant Scheme for Internet Voting (Extended Abstract)	330
<i>Roberto Araújo, Sébastien Foulle, and Jacques Traoré</i>	
Scratch, Click & Vote: E2E Voting over the Internet	343
<i>Miroslaw Kutylowski and Filip Zagórski</i>	
Securing Optical-Scan Voting	357
<i>Stefan Popoveniuc, Jeremy Clark, Richard Carback, Aleks Essex, and David Chaum</i>	
Attacking Paper-Based E2E Voting Systems	370
<i>John Kelsey, Andrew Regenscheid, Tal Moran, and David Chaum</i>	
Aperio: High Integrity Elections for Developing Countries	388
<i>Aleks Essex, Jeremy Clark, and Carlisle Adams</i>	
Author Index	403

The Witness-Voting System

Ed Gerck

Safevote, Inc.

P.O. Box 9765, San Diego CA 92169, USA

egerck@safevote.com

<http://safevote.com>

Abstract. We present a new, comprehensive framework to qualitatively improve election outcome trustworthiness, where voting is modeled as an information transfer process. Although voting is deterministic (all ballots are counted), information is treated stochastically using Information Theory. Error considerations, including faults, attacks, and threats by adversaries, are explicitly included. The influence of errors may be corrected to achieve an election outcome error as close to zero as desired (error-free), with a provably optimal design that is applicable to any type of voting, with or without ballots. Sixteen voting system requirements, including functional, performance, environmental and non-functional considerations, are derived and rated, meeting or exceeding current public-election requirements. The voter and the vote are unlinkable (secret ballot) although each is identifiable. The Witness-Voting System (Gerck, 2001) is extended as a conforming implementation of the provably optimal design that is error-free, transparent, simple, scalable, robust, receipt-free, universally-verifiable, 100% voter-verified, and end-to-end audited.

Keywords: voting, trustworthiness, secret ballot, error-free.

1 Introduction

It is known that current voting systems when applied to public elections consistently produce results that are untrustworthy [1–3]. Centuries of experience with paper ballot voting, decades of experience with the computerization of election-related functions and with electronic ballots have not significantly altered this picture [4–8].

Many blame the secret ballot [9] requirement as posing an impossible problem to solve. Rather, such examples, together with the unsuccessful attempts to improve election outcome trustworthiness, suggest that there is today no effective model of how information should be collected and handled in a realistic voting system environment that includes faults, attacks and threats by adversaries.

¹ A secret ballot (voter privacy) is commonly used to prevent voter coercion and vote buying. Voter privacy is legally protected in many jurisdictions. For example, a provision of the US Washington State Constitution states: “secure[s] to every elector absolute secrecy in preparing and depositing his ballot”.

We raise this conjecture in order to show the need for a new, comprehensive voting process model that can be used to explain the observed behavior with any type of voting process (with or without ballots) in the presence of faults, attacks and threats by adversaries. We want the model (hereafter, the *Voting Information Transfer Model* or VITM) to predict potential areas of improvement with an effective design that improves election outcome trustworthiness. We further want the model to be promotive of voting system requirements (hereafter, the *Requirements*) that include the secret ballot and allow a conforming voting means (exemplified by the *Witness-Voting System* or WVS) to be developed.

Since the assertions of any such model have to do with the relationships between information elements such as *sender, recipient, encoder, decoder, messages, and interference* representing, for example, a voter casting a ballot in the presence of an adversary, the VITM is based on Shannon's Information Theory [9-13], where information is essentially stochastic in nature. We posit that insufficient consideration of this circumstance lies at the root of the difficulties with voting systems at this time.

In other words, after centuries of experience with paper ballot voting and decades of experience with computerized and electronic voting, we reached what we could call a *classical barrier*². If we want to make progress, we cannot continue to treat voting information *classically*, although the ballot cast by a voter is and remains *deterministic*.

1.1 Outline

Section 2 is an informal presentation of our approach with a first, highly simplified, WVS implementation.

Sections 3 and 4 discuss current problems and previous work, for both paper ballots and electronic voting, focusing on voter privacy and election outcome trustworthiness.

Section 5 presents the intuition that, although voting is a deterministic process, in order to qualitatively improve how to best cast and count ballots in the presence of faults, attacks and threats by adversaries, we will have to look further into the information flow using Information Theory.

The three components of our framework are the model, the requirements, and the conforming voting means, as respectively defined:

Voting Information Transfer Model (Section 6)
 Voting System Requirements (Section 7)
 Witness-Voting System (Section 8)

Section 8 includes the detailed presentation of a second WVS implementation, extended from our first results [14] in 2001. In the Conclusions, we discuss extensions and applications. We note that this work is applicable not only to voting *per se* but also to voter registration and other aspects of the voting process that are relevant to voter privacy and election outcome trustworthiness.

² Hereafter, the term *classical* indicates that information is treated deterministically.

2 Informal Description - The HS/WVS

A highly-simplified *Witness-Voting System* example is presented in Table 1. An Election Operator (EO) manages the HS/WVS, which includes special elements called *witnesses* and *readers*. For example, witnesses capture the ballot as seen and cast by the voter, which are later tallied by readers. Witnesses and readers are placed at least with triple redundancy, so that differences can be resolved and no single failure could compromise error detection. Witnesses' recordings are inaccessible until the election ends. The EO has previously run a voter registration service (where the WVS can also be applied), which generated a list of eligible voter authorizations to allow access to a ballot.

Table 1. A Highly-Simplified WVS (HS/WVS)

<i>Voting System Setup</i>	The Election Operator (EO) sets up a precinct, where an empty ballot box is placed, with video cameras watching 24/7 and recording every step of the voting process, as EO witnesses.
<i>Independent Verification Setup</i>	Central to the WVS method, the EO invites each stakeholder to add their own witnesses. More Witnesses = Better Evidence. Stakeholders can also add their own readers, e.g. in providing a transparent <i>error-detection and consensus process</i> for tallying.
<i>Election Run</i>	All the witnesses are able to see and record that the ballot box is empty before it is sealed and the election starts. All as witnessed at each step, anyone who comes to vote is verified for eligibility, signs the voter list and receives a ballot, marks the ballot secretly, and inserts the voted ballot into the ballot box. At the end of the election, the ballot box is opened and the ballots, including all the ballot images from the witnessed records, are used in a transparent <i>error-detection and consensus process</i> to arrive at a high accuracy, high reliability and trustworthy election tally.

The WVS design invites stakeholders to be part of the election setup. For example, political parties may add their own witnesses and readers, that they can trust and verify. This design consideration is critical to the trustworthiness of the election's outcome and assures transparency. Since *transparency is in the eyes of the beholder*, we let all be satisfied with it by construction.

Stakeholders can make sure that each vote is counted as seen and cast, but the secret ballot is not compromised. Various witness, provided by the EO and stakeholders, anonymously record each ballot as seen and cast by voters (thereby catching the expressed voter intent). Ballot information from each witness and the ballot box is used by readers in an *error-detection and consensus process* to transparently calculate the election tally where, to further protect voter privacy, individual ballot choice patterns do not need to be disclosed.

Witness and reader elements are verifiable to be free from errors, but there is no requirement for all elements to be perfect or even perfectly independent. *Perfection of each human and each element of hardware and software is not required.* In fact, we know that *all* elements are somewhat imperfect. The security paradigm that the weakest link defines the security of the system does not apply. Rather, a central aspect of the WVS is that there should be enough ³ *multiple correction channels (C)* providing feedback in order to enable the WVS to offset the influence of interference from error channels (*E*) caused by faults, attacks and threats by adversaries, so that the election outcome error can be reduced to a value as close to zero as desired, which we call *error-free*.

In other words, the WVS can achieve an error-free election outcome by optimally preempting, or at least resolving, any dispute regarding *accuracy, reliability, voter privacy, and election outcome trustworthiness*.

2.1 Trust Is Good, Control Is Better

Trust can be viewed as that which can break a security design [15]. In other words, when I trust A on matters of X, if that trust fails then I have to assume that “matters of X” can take on any possible value. With possible exceptions, it is better to control than to trust [15]. Thus, with the WVS, no one is asked to trust a particular witness, reader, or even a particular procedure.

To comply with these goals, the WVS allows witnesses and readers to be independently controlled by each stakeholder, so that they do not need to be trusted by that party. Cryptography is not used in any role where it *must* be trusted by a party —trust cannot be imposed [15]. Cryptography can, however, be used where each party agrees to it, for example in using public-key cryptography to protect the information collected by that party’s witnesses in such a way that the resulting information confidentiality is acceptable by any party. Alternatively, acceptable physical controls can be used.

2.2 Concept Appeal

Intuitively, as more witnesses are considered, it becomes less likely that all witnesses can be compromised at the same time. More Witnesses = Better Evidence. The idea that *multiple correction channels* can be used to offset errors caused by fraud was already known some 500 years ago in the context of combating corruption.⁴

Formally, the WVS and the *error-free* result are based on the well-known Information Theory [9, 11], a mathematical representation of the conditions and parameters affecting the transmission and processing of information, which has been applied as a natural answer to questions in fields as diverse as cryptography and linguistics [10], optics [12] and portfolio theory [13].

³ Qualified as $C \geq E$, see Section 6.3, Error-Free Condition.

⁴ Hindu governments of the Mogul period, notwithstanding the additional efforts, used at least three parallel reporting channels to survey their provinces with some degree of reliability [16].

2.3 Questions and Answers

Q1: What happens if a witness element malfunctions, is compromised, or records are erased, or edited?

A1: *The witnesses are placed at least with triple redundancy. Because each witness is at least somewhat independent, the chance that a fault or attack will affect $N > 1$ witness at the same time is a decreasing function of N . More witnesses can be added, as needed.*

Q2: What happens if the number of voters who signed the voter list is not exactly the same as the number of ballots in the ballot box?

A2: *A correct tally result can be achieved when the various witness ballot recordings are played and used in the transparent error-detection and consensus process. In addition, witnesses can also be used to verify the voter list.*

We propose Q3 and Q4 to further explore the WVS design.⁵

Q3: How do I know that when I selected and cast a ballot for candidate A, that something hidden in the ballot box did not change my vote to B?

Q4: If my vote is supposed to stay secret, how can I verify that it was counted correctly?

3 The Problem

Election outcome trustworthiness has been a longstanding problem. To exemplify the problem setting, consider a typical example where a known number (who sign the voter list) of known voters (who are identified in voter registration) receive pre-approved ballots, privately make choices on race options, and cast their ballots into a one-way ballot box; after the election is over, the ballot box is opened for the first time and all ballots are tallied, with totals made public. No one knows how anyone else voted (the secret ballot condition).

The equation:

$$\text{Number}(\text{Voters}) = \text{Number}(\text{Ballots})$$

represents the basic and intuitive requirement that the *number of voters who voted must be equal to the number of ballots in the ballot box*. In the US, today, this requirement is explicitly not used in public elections with any voting system, paper or electronic. The official reason given for this omission is that it is impossible (*sic*) to do so with secret ballots, as no one knows which ballots may be extra or missing in case of a difference.⁶

The official procedure in such cases is to *count all the ballots found in the ballot box*, even though the signed voter list may contain a different number of voters. In support of this procedure one often hears the argument that if the number of extra or missing ballots does not influence the election outcome, no

⁵ For comments, please use this paper's full title in the Subject line of the email.

⁶ We note that the HS/WVS uses secret ballots *and* solves this problem; see Q2/A2.

one should care —even though undesirable ballots may have been taken from the ballot box or desirable ballots may have been inserted.

However, when a voting system allows even *one* ballot to be missing or to be inserted undetectably, this is sufficient proof that no ballot can be trusted. More strongly stated, although *no* count difference may be found between the voter list and the total of ballots in an election, still any or all the legitimate ballots may have been compromised—for example, substituted 1:1 with ballots that were forged, tampered with, or falsely invalidated.

Yet if ballots were not changed at all, as proved by digital signatures or any other mechanism, votes may still have been marked not in the way that they were seen or were intended by the voter (e.g., a vote for a candidate is shown to the voter on the screen and matches what is also shown on a voter-verified printout, but the vote is not recorded for that candidate), votes may have been tallied incorrectly, or votes may have been revealed before the tally⁷.

Even if vote tampering *does* seem to have occurred and influenced the election outcome, the credibility afforded to pre- and post-election Logical and Accuracy (L&A)⁸ tests and other contributing factors such as cost to rerun an election and legal statutes protecting trade secrets, have such a force in the balance that a candidate may not prevail in challenging the election outcome, albeit all the contrary evidence that the candidate can collect⁴.

However, satisfying an L & A test does not mean that the voting process is trustworthy in the presence of an adversary that would interfere with the voting process during the election (but hide its influence during L & A tests). Further, a zero count difference when repeatedly reading votes from the same stack of ballots does not mean that the ballots were the same that voters saw and cast.

To shed light into the problem of vote counting errors, it is important¹⁷ to distinguish *accuracy* from *reliability*, as shown in Table 2. See Fig. 6.5 in¹² and Slide 3 in¹⁸ for a visualization of these definitions. As Table 2 shows, an L & A test fails to effectively measure either accuracy or reliability.

Table 2. Measurement Error Type Definitions

<i>Accuracy</i>	The spread in measuring a single event. For example, whether a vote that was seen and selected to be cast by a voter can be counted or not from a ballot.
<i>Reliability</i>	The degree to which measurements are expected to be replicable in varying circumstances yielding the same results. For example, whether the same votes cast before and during the election provide the same tally result.

⁷ Vote leakage can guide get-out-the-vote and voter-suppression operations.

⁸ A Logic and Accuracy test is a deterministic test performed in a controlled environment. It consists of running sufficiently large predetermined patterns through the voting system, capturing ballot images, and tabulating the ranked choice results. The ballot images and the tabulated results can be compared to the predetermined patterns.

4 Previous Work

The effective use of computing technology has been promoted by Saltman [19, 20] and others for more than 40 years.⁹ The application of cryptography dates from the early 80's (Chaum [21]).

A well-known approach to improve election outcome trustworthiness is to add auditing mechanisms¹⁰ with the intent to preempt or at least resolve a dispute regarding the proper casting and counting of votes. For example, to provide an acceptably high confidence level that all ballots were counted as cast.

Auditing proposals usually differ in their methods as applied to electronic voting and paper ballots. Particularly relevant to the use of computers [2, 8], auditing should be understandable by voters.

A number of independent verification mechanisms and verification enhancements, including voter-verified and universally-verifiable methods, have been proposed for example by Cohen and Fischer [22], Benaloh [23], Mercuri [24], Cramer and Franklin [25], Benaloh and Tuinstra [26], Gerck [14, 17, 18, 27], Neff [28], Jakobsson, Juels, and Rivest [29], Kiayias and Yung [30], Mercuri and Neumann [31], Chaum [32], Chaum, Ryan and Schneider [33], and Chaum et. al. [34]. Other contributions include auditing systems for the software used in elections, for example from Garera and Rubin [35], as well as proposals calling for using more-easily-auditable open-source software such as by Wyson [36], Kitcat [37], and the Caltech/MIT Voting Technology Project [38].

Auditing should also allow recounting, to duplicate the result of an election. The purpose of a recount is to correct or confirm the results. California, for example, has mandated a 1% recount of all ballots automatically without payment from a candidate. A recount, however, is critically flawed in terms of auditing if *only* the same elements are counted again, since there is no independent source to verify them (see footnote 10).

As reviewed in the next two sections, a common limitation in *classical* paper ballot and electronic voting systems is in providing the needed audit capabilities of voters and ballots while still satisfying voter privacy (election best practice rules and US state laws require a *secret ballot*, see footnote 1).

In particular, voter-verified auditing should not cause a privacy violation problem. A voter who discovers an error ought not lose the privacy of voting in the course of the demonstration of an inconsistency. The *secret ballot* requirement also fails for a voter-verified or universally-verified auditing method when voter privacy is protected by trusting a *quorum* of verifiers or election operators, with a *threshold of collusion*, as discussed in Section 7.1, under “Computational Privacy”.

⁹ Computerized voting with punched cards was used in 1968 in Los Angeles County, Calif., USA [19].

¹⁰ It is well-understood that a voting system should be auditable. An audit is an independent verification; it must be carried out in ways that are significantly different from the initially accomplished task, including the use of different machines and people.

4.1 Electronic Voting

The state-of-the-art, and recently a legal requirement in some jurisdictions, with electronic voting is called voter-verified audit paper trail (VVAPT) [24, 31]. The VVAPT is a printout of the final voting screen with all the votes confirmed by the voter.

The purpose of the examination of the VVAPT by the voter is to verify that the selections shown on the VVAPT are identical to the selections shown to the voter on the final voting screen. However, unknown sampling by voters (it is unknown whether enough voters verify) and violation of voter privacy to report a problem (see end of Section 4) have been noted [19]. This method also eventually discloses all the cast ballot choices, which is vulnerable to “*voter pattern fingerprinting*” (see Section 7.1).

More critically, and in spite of its name, the VVAPT *does not allow the voter to verify* that the vote was stored correctly. This denies reliance *on the voter* in detecting a malfunction as a way to prevent fraud—a programmer can make the printout and the screen seen by the voter coincide exactly, and yet a different result is stored for tallying.

The VVAPT may be used in a *hand recount*, or as the actual ballot for *hand counting* the votes (the machine count would just provide a knowingly-unreliable indication of the vote count). However, if all that one has is a paper ballot, it may make more sense in terms of cost, time, human errors and fraud prevention factors to use optical-scan paper ballots in the first place.

4.2 Optical-Scan Paper Ballots

The state-of-the-art in paper ballots is the optical-scan paper ballot. Some participants in this dialogue consider that there is no better way to vote. For example, California Secretary of State Debra Bowen said in a Keynote address at USENIX 2008 that “*Voting and counting paper ballots are things that all citizens can understand and in the case of random hand tallies, something that all citizens can observe and understand*”. [39]

There are two types of optical-scan ballots: voter-filled and machine-printed [19]. In either case, a voter is able to visually verify the voted ballot before casting the vote, which may be done for example by postal mail or by inserting it in an optical-scan unit. A visually-impaired voter may not be able to read or mark the ballot, but that type of voter may use an audio assist unit.

Optical-scan ballots may be recounted by hand or on an independently-managed computer system, and thus are considered adequate [19] to provide the basis for a recount, either partial or full, to check the initially reported results.

We question this conclusion for a number of reasons.

First, the consideration that optical-scan ballots can be used as the auditing source for the ballots themselves, even if recounted by hand or on an independently-managed computer system, is at odds with the basic principle in auditing —*independent verification*. A record cannot be used to audit itself. A

hand recount or a machine recount of the same optical-scan ballots should not be considered an independent recount or a satisfactory auditing of the process. If we play the same CD in different machines, we still hear the same songs.

An optical-scan system may count and recount exactly the same number of false votes, since anyone can internally mark any choice in an undervoted¹¹ race before the first count.¹² Such change would not involve any ballot swap and would be undetectable in a visual inspection or a machine recount. Further, ballots may be swapped, remarked, reprinted or exchanged during scanning (scan a different set). Although a voter may see that his voted ballot was scanned and verified as valid, the ballot can later be fraudulently overvoted¹³ to invalidate it, when the voter is no longer in sight. Again, a hand recount or a machine recount of the same optical-scan ballots would not resolve these issues.

Optical-scan ballots are sensitive to stray marks and may count them as a vote—for example, if a voter accidentally pauses with a pen over an oval and then decides not to vote for any candidate, that mark may be counted as a vote. Use of a non-standard pen by the voter, or by the voter indicating a choice with a mark that is not readable by the scanner’s sensor, or by a degraded sensor that is not reading correctly, or a ballot that has been contaminated with smudges or bending, may also change the intended votes readable on the ballot. Hand reading the optical-scan ballot may resolve the issues mentioned in this paragraph.

On the topic of voter privacy, optical-scan ballots disclose all the cast ballot choices, which is vulnerable to “*voter pattern fingerprinting*” (see Section 7.1). Further, voters may use a pen with invisible ink to identify their paper ballots, which marks can then be read by someone, even well after an election, in order to reward or punish voter behavior (e.g., tagged voter did not vote as expected). Forcing voters to publicly verify the correctness of their own ballots by scanning them in public, or by having their ballots printed by a machine, is also a potential violation of voter privacy. Voters should be able to express their disagreement with the election choices, for example by overvoting, undervoting or by simply nullifying their ballots (e.g., by writing on a paper ballot), without coercion.

Adding a voter-verified or universally-verifiable auditing enhancement to optical-scan voting systems would be useful to introduce an auditing record that is not the ballot itself.

However, a voter-verified enhancement for optical-scan voting systems implies a privacy violation problem (see end of Section 4). The *secret ballot* requirement also fails for auditing enhancements in optical-scan voting when voter privacy is protected by trusting a quorum of verifiers or election operators (see end of Section 4).

¹¹ An undervote means to make less choices than possible in a race; if no choice is made it is also called a blank vote. Voters may want to undervote.

¹² Even if there is a specific oval for a “No Vote” or “Abstain” choice, voters may still just mark the top races and not mark anything else down the ballot.

¹³ An overvote means to make more choices than what is allowed in a race; it nulls an evaluation of that race (it becomes indeterminate). Voters may overvote as a protest.

5 Intuition

What if perfect human clerks¹⁴ who are ideally honest and error-free ran an election, would the result be trustworthy?

No, not necessarily. Trustworthiness of the election outcome would still depend on whether a number of requirements are met, such as that no one can vote more than once or vote on behalf of another. This example refutes the oft repeated idea that *the trustworthiness of an election is entirely dependent on the people who count the votes*. Even if the people who count the votes are ideally honest and use flawless devices with flawless software, the trustworthiness of the election still remains elusive.

Election outcome trustworthiness depends on voters, but not only on their honest behavior. It also depends on voters understanding the instructions and being able to let their intent be represented accurately and reliably in the votes they cast, unlike with the notorious Florida “butterfly ballot”.

What if the election is not run by such ideally honest human clerks or computers? We expect that *additional* requirements would have to be imposed in terms of election outcome trustworthiness. We may require that ballots must be handled in the presence of at least two clerks. But, would two clerks allow “enough” risk reduction to allay fraud concerns? What do we mean by “enough”?

Many will recognize that not only there is no good model to design these important requirements to, and one must make up the rules by trial and error, but there is no inner metric to comprehensively define “enough” in terms of risk for each step and for the voting process as whole.

What we are missing is a better model. The deterministic model described by “*cast ballots, then count them*” fails to provide us with guidance for improving the trustworthiness of the election outcome.

Instead, we start with the fresh observation that *a voting system is an information transfer system*. The tally results contain information that was sent from each voter, where, according to Information Theory [9, 11], information is essentially *uncertain* in nature —although voting is *deterministic*.

Uncertainty which arises by virtue of freedom of choice on the part of the voter is desirable uncertainty. Uncertainty which arises because of interference on that freedom of choice (e.g., caused by faults, attacks and threats by adversaries) is undesirable uncertainty, which we call errors. We are, thus, motivated to include the possibility of errors by means of a probabilistic description of undesirable uncertainty. Information Theory provides a good correspondence here as well, where the undesirable uncertainty is called *interference* (or *noise*).

In such a model, voting system requirements are not arbitrary. *Requirements are created and dictated by the goal of minimizing interference*. The intuition is that an information-theoretical voting model should be able to optimally combat interference and, thus, improve the trustworthiness of the election outcome.

¹⁴ Or perfect *computers*. Originally, and as late as the 1920s, *computers* were defined as human clerks who performed computations. *Computers* were expected to perform obediently with paper and pencil, for as long as it was necessary, but without insight or ingenuity.

6 The Voter Information Transfer Model (VITM)

The VITM comprises the voting process in its general aspects. We consider a set of voters with access to *voting means* controlled by an election operator (EO). The *voting means* is the totality of physical means used, such as electronic or human-based, with ballots or not, to collect the voters' expressed intent and provide an election outcome with a tally of votes for each respective race in the election. No one knows how anyone else voted (voter privacy), not even the EO. Different times and places may be used for voting. Voters may see different options; e.g., voters may be presented with different races (e.g., using ballot styles), with a different option order for each race (e.g., using ballot rotation), use different languages and media (e.g., touch screen, voice). We include the possibility of faults (from various sources including hardware, software, and human error), attacks (passive and active) and even just threats by adversaries, all of which can *interfere* with the voting means and with the voting process, with varying consequences including, most notably, to influence the outcome.

By direct correspondence with Information Theory concepts [9, 11], we formally identify signals with the votes as seen and cast by each voter, which are encoded and sent by a communication channel (e.g., a ballot), and are received at a relay point (e.g., the ballot box), where they are combined with other inputs and eventually decoded (tallied) to produce the output signal (the election outcome). The signals are selected by each voter within a number of options that includes all possible signal combinations, as defined by the election's rules for that voter. The voting system must be designed to operate for each possible selection (e.g., voted ballot), as voters must have freedom of choice. Faults (e.g., human, software), attacks and threats by adversaries represent interference that may change the output signal (causing election outcome errors).

To prevent any confusion, it must be emphasized that *the votes seen and cast by a voter are not stochastic variables in our approach*. The fact that Information Theory uses probability distributions to describe signals as chance variables does not mean that we are modeling the cast votes as randomly changing their nature between states, or as a random superposition of states. Rather, given the available evidence (e.g., ballot box and witness records), we use probabilities to represent degrees of *belief* (belief is the probability that the evidence supports the claim) about the mutually exclusive hypotheses as to what the cast votes might be, of which *only one* set of selections (cast ballot) is actually true for each voter. This represents the condition that only one ballot is valid per voter¹⁵.

In a communication system, the message is what is transferred from sender to receiver. According to Information Theory, the message has *information* measured by the uncertainty as to what the message may be. The cast votes have *information* because while the votes are chosen among a number of known choices, the selection is unknown except by the voter (which satisfies the secret ballot condition). The observation that the voter knows the selection is not relevant;

¹⁵ Even if each voter is allowed diverse opportunities to vote (e.g., to prevent coercion online [18]), there is only one set of selections that is true for each opportunity.

the relevant observation is that, at the time the vote is cast, the corresponding output signal (the contribution of the vote to the election result) is unknown at the end point.

Thus, both conditions of assuring only one valid ballot per voter and a secret ballot occur naturally in the VITM; yet we recall that they present difficulties in *classical* voting systems.

These initial observations, further qualified by us elsewhere [14, 16–18, 40, 41], make Information Theory a natural candidate for modeling the voting process. Mathematics of an ever more elaborate variety is necessary, but to better focus on our search for more refined concepts, we now claim a metamathematical argument and directly use well-established results from Information Theory in terms of the Voter Information Transfer Model.

6.1 Limitations

In our approach we apply Information Theory concepts and results to physical signals in communication channels and elements that we identify. By extension, we shall also apply Information Theory to some *conceptual* signals and communication elements, which are observable in their effects on the election outcome of the physical device and yet are non-physical, as they originate from environmental or non-functional influences. For example, in section 6.2 we consider a *conceptual* coercion channel where an adversary may try to use threats or rewards in order to perturb how a voter makes ballot choices.

We expect that the Information Theory results extended to those conceptual signals will generally hold true in the restricted context we use them, not only given the broad application of Information Theory in many fields [10–13] but also given our argument (to follow) that any conceptual interference of concern here must be physically observable in its effects.

However, the only formal claims of this work are made for physical signals, such as with the voting means.¹⁶

6.2 Interference

To further investigate the requirements of voting system design in providing election outcome trustworthiness, we now divide the actual election outcome in two parts: the ideal part, without any interference; and the interference. We shall call the ideal part the *election outcome* and consider the actual case to be that where the *election outcome* is perturbed by the *interference*. Interference is undesirable uncertainty. Interference causes errors.

*Interference is anything that can change the election outcome compared with what the election outcome would have been if the interference did not exist.*¹⁷

¹⁶ The Witness-Voting Systems (WVS) of Sections 2 and 8 are examples of a voting means. They are physical devices with physical signals (e.g., a voted ballot).

¹⁷ Note that we explicitly exclude from this definition of interference any perturbations that have no influence on the election outcome. This is not necessary. Interference could be defined as a vector quantity with other components.

We further distinguish interference with functional and performance influence (hereafter called *physical interference*) from interference with environmental and non-functional influence (hereafter called *conceptual interference*).

Conceptual interference must be observable in its effects but does not need to exist physically; it may just stay as a threat.¹⁸ Interference also presents combined failure modes where an attack in one layer of the system can be used to compromise another layer (e.g., a *conceptual* interference creating a *physical* spurious change in the election outcome).

Our concept of interference captures any source that could perturb the election outcome, including faults, attacks and threats by an adversary.¹⁹

For example, passive eavesdropping on the voted ballot (e.g. by covertly monitoring the stray electromagnetic emissions from the computer screen used by the voter) can enable coercion that may interfere with the election outcome. Yet if performed from afar, undetectably and never overtly used to coerce or influence voters, a passive attack such as eavesdropping can still be used to perturb the election outcome (see footnote 7). In either case, passive eavesdropping can be modeled as an interference source in terms of its *influence* on the output signal (the received message; the election result).

Information Theory also includes the concept of interference, or *noise*, defined as that which perturbs the signal. The usual case of interest is when the signal does not always undergo the same change in transmission, when noise may be considered a chance variable just as the message is considered. In general, noise may be represented by a suitable stochastic process. However, it matters not whether noise always produces different changes in the received signal, or where that change originates. A constant and 100% predictable radio signal from an unknown source is also noise. Anything that interferes with the message is noise.

We observe that interference (noise) in Information Theory corresponds to the same concept defined here. As previously considered by us [16], the condition to model attacks and faults as interference (noise) in a communication system is the same one that already exists in Information Theory, namely, noise is anything that interferes with the message.

In describing interference sources and prevention, it is customary to define boundaries or spheres of influence. A first boundary comprises the voting means.

¹⁸ This is well-known to chess players, where perceived threats can be more effective to change a game than actually carrying out an attack. A voter who fears that an attack can reveal her choices, with unpleasant consequences, may not vote as intended even if there is no such attack.

¹⁹ Including, for example, the influence of ambiguous ballot design, incorrect touch screen coordinates, transmission and reception errors, faults, malfunctions, virus, bugs, buffer overflow, dormant or hidden code, alpha particle memory corruption, covert microcode, covert channels, human error, collusion, coercion, blackmail, financial kickbacks, fraud and any passive or active interference attempt by adversaries. This may also include man-in-the-middle, eavesdropping, replay, impersonation, forgery, and any other attacks by an adversary. Attacks may also adapt to defenses, either automatically or driven by an intelligent source [16].

A second boundary (which contains the first) comprises the voting system with the voting means, voters, election operators, and ancillary machines such as those used for voter registration. A third boundary (which contains the second) is open and includes anything else that lies outside the second boundary.

The VITM and the Requirement operate in the third boundary and define how, inside the first boundary, a conforming voting means (e.g., the WVS) operates.

Inside the first boundary (the WVS), interference prevention must be limited, of course, by physical signals. This may seem at *first sight* to present a basic security limitation for the VITM design, as interference sources from outside the first boundary may still perturb the voting means operation.

For example, outside the first boundary but still *inside* the second boundary, lie physical and conceptual interference sources that do not seem to be preventable by the voting means (e.g., collusion between election operators; physical threats against voters who do not vote as ordered; malicious code inserted into the voting means by an adversary).

Moreover, significant interference sources lie even *outside* the second boundary. For example, *gerrymandering*²⁰ and selective “voter roll purging”, or just conveniently ignoring an existing imbalance of natural factors (e.g., illiteracy, language differences, economical handicap, and physical handicap) that can block undesired voters or favor participation of desired voters.

However, contrary to our concern at *first sight*, it is easy to show that conceptual interference can be physically prevented by a conforming voting means (the WVS). For example, an attempt by a voter to sell the vote cast (conceptual interference) can be physically prevented if the voting means conforms to a requirement to be receipt-free (the voter cannot prove to others how she voted). Conversely, a physical control vulnerability at the voting means may open the possibility of conceptual interference. Even if a voter just fears that voter privacy can be compromised by some characteristic of the voting means (e.g., perceived lack of a physical control preventing “voter pattern fingerprinting”, see Section 7.1), the voter may not vote freely (conceptual interference).

More generally, we argue that a conceptual influence must eventually create a physical influence in order to be an interference source.²¹ Thus, the respective pairs of conceptual and physical influence are not independent and can, if desirable, be controlled (denied or allowed) physically. Conversely, ignoring such considerations in the voting system design could create conditions for unintended conceptual and physical interference.

²⁰ This term describes the deliberate rearrangement of geographical limits of congressional districts to influence the outcome of elections. Its purpose is to concentrate opposition votes into a few districts to gain more seats for the majority in surrounding districts (packing), or to diffuse minority strength across many districts (dilution).

²¹ In other words, it must be observable. If no physical influence is created (i.e., if there is no change of election outcome), by definition the interference does not exist.

Thus, even though useful to define the physical scope of each element, *boundary definitions are not limiting in terms of either interference influence or prevention.*

Our framework [VITM, Requirements, WVS] predicates the need for a comprehensive, cross-boundary approach. In order to minimize interference, we consider conceptual and physical error/correction channels in the VITM to define Requirements (Section 7) that work together with a conforming voting means (the WVS, Section 8)²²

6.3 Optimal Design

Election outcome trustworthiness requires that a voting system produces results with high *accuracy* and *reliability* (see the respective definitions in Section 3, Table 2). This section shows how 100% accuracy and reliability can be achieved and verified as closely as desired, which we call the *optimal design*.

From Information Theory we use the concept of *channel*, as that part of a communication chain in which signals are transmitted from a sender to a receiver. An important channel in a voting system is shown below, in sending information from A to B:

$$(A: \textit{what the voter sees and casts}) \implies (B: \textit{the tally results})$$

where, if the channel is vulnerable to interference, it may not be possible to reliably send A to B. The *fundamental problem of voting* [18] is that of reproducing at B the same information that was sent from A. Or, as often stated, how can we prove that the vote received at a ballot box, and tallied, is the same vote that was seen and cast by a voter?

This question is not easier to answer if the voter is close to the ballot box, or far away. Distance plays no role, contrary to what one might think at first. The essential problem is that the voter is not *and cannot be* inside the ballot box, and cannot follow the ballot all the way to the tally results, hence the voter has no way of knowing if what was sent through that communication channel (which may be very short) was what was received (and tallied).

Specifically, what we desire is stability in the presence of interference meaning, in a broad sense, “invulnerability to potentially corrupting influences”. This corresponds to the concept of *reliability* in Information Theory, with the same definition used here (Section 3, Table 2).

²² For example, gerrymandering is a conceptual attack that can be set up in spite of any security assurances of the voting means. However, if a conforming voting means allows voting from any location then it becomes possible to deny in the requirements, as broadly as desired, any geographical restrictions that could be manipulated for gerrymandering [42]. As another example, visual accuracy errors by the voter in choosing the intended ballot options (i.e., a conceptual interference) can be prevented physically by offering the voter a final screen with a summary of all the votes for confirmation.

To increase reliability *in spite of* interference, Information Theory introduces the idea of using different channels of information as intentional *redundancy*²³. More channels, more redundancy, less interference.

It is clear from the foregoing, thus, that by sending information from A to B in a properly redundant form, the probability of accuracy and reliability errors can be reduced—which can be theoretically and experimentally verified. For example, by sending to B two different messages created from the same final ballot screen that the voter saw and confirmed: (1) a printout and (2) an electronic record. With more properly redundant channels (e.g., a copy of the screen memory), the probability of errors could be made even smaller.

One could expect, however, that to make the probability of errors approach zero, redundancy must increase indefinitely, and the rate of transmission therefore would approach zero. This is by no means true, as shown by the Tenth Theorem (in current wording) [9, 11]: *With the addition of a correction channel equal to or exceeding in capacity the amount of noise in the original channel, it is possible to so encode the correction data sent over this channel that all but an arbitrarily small fraction of the errors contributing to the noise are corrected. This is not possible if the capacity of the correction channel is less than the noise.*

By direct application of the Tenth Theorem, we state below the condition to make the probability of errors approach zero.

Error-Free Condition: There exists an *optimal design* that can reduce election outcome errors (interference) to a value as close to zero as desired, which we call *error-free*. The existence condition is given by

$$C \geq E$$

where C is the capacity of the correction channels and E is the capacity of the error channels.

Referring to the Intuition discussion in Section 5, not only can we now quantify what we mean by *enough*, as $C \geq E$, but we can also specify in terms of inner metrics the expected risk value for each error channel and for the voting process as a whole.

The *optimal design* is used in our framework to reduce both physical interference (provably) and conceptual interference within all three boundaries (see boundary definitions in Section 6). In physical terms within the first two boundaries, a conforming voting means implements the *optimal design* to provably increase accuracy and reliability in a communication process that includes the ballot as seen and cast by a voter (the starting point A), the ballot box as a

²³ *Redundancy* is the variety in a channel that exceeds the amount of information actually transmitted. English writing is estimated to be 50% redundant, which accounts for our ability to detect and compensate for typing errors; fr xmpl, ndrstrndng txt even without vowels. In the process of communication, redundancy is essential to combat interference, to assure reliability and to keep a communication process in operation in spite of interference [9, 11].

relay point, and the tally results (the end point B). The capacity of the correction channels physically available to the voting means are set and adjusted²⁴ to achieve the condition $C \geq E$, whereby the election outcome errors (interference) can be as close to zero as desired. The Requirements composition and the implementation development (including software) are similarly set and adjusted, where correction channels are provided by the EO, the stakeholders, the voters, and other elements within the third boundary.

The significant aspect is that the election outcome is error-free. It may seem surprising that we should define an error-free result for a voting means in the presence of interference, including the possibility of errors caused by faults and attacks by adversaries, since we do assume in such circumstances that the cast ballot may have been changed from what the voter saw and confirmed. However, rather than strive for elusive perfect elements that could outright eliminate errors as in a *classical* deterministic model, our approach is to explicitly include the possibility of errors by means of a probabilistic description of undesirable uncertainty (interference), where the Error-Free Condition is then applied.

6.4 Trust

We now impose what we call the *transparency* condition, in that to allay collusion and security concerns we want the error-free condition to be publicly verifiable.

However, not all messages in a voting system are capable of providing acceptable proof to any stakeholder as a verifier. For example, a proof that may be acceptable by the EO may not be acceptable by voters. In order to rate sources, destinations, and communication channels in terms of providing acceptable proof to a verifier, the concept of qualified reliance on information is introduced in the VITM based on our previous definition of trust in a communication process (Gerck, 1997 [15]), which is compatible with Information Theory.

According to [15], *trust* has nothing to do with feelings or emotions. Trust is communicable. However, trust cannot be communicated by self-assertions (e.g., saying “Trust me” does not make one more trustworthy). Formally stated, *Trust is that which is essential to a communication channel, but cannot be transferred using that channel*. From this abstract definition, applied definitions can be derived such as *Trust is expected fulfillment of previously observed behavior* and *Trust is qualified reliance on information, based on factors independent of that information*. In short, trust is defined by at least two channels of communication and the channels need to be at least partially independent. In plain English, the greater the number of independent ways one can verify something, the greater reliance one may have.

Section 6.3 showed that with more channels, more redundancy, we can reduce interference. Now in terms of a trust requirement, the desirability of multiple communication channels again enters our framework.

Multiple communication channels can use any media for information transfer, such as electric signals, magnetic and optical disks, paper, and microfilm.

²⁴ For example, by using operational feedback; see Table 4, Section 8.

However, suppose that a terminal where the voter enters his choices would change them to something else and then would send this corrupted information over N channels using diverse media. Does it make any difference in terms of trust whether $N = 1, 2$ or 500 ?

Not in terms of trust on the cast ballot. In such a case N would still be 1 for the ballot channel. The 2 or 500 channels are not independent for the ballot channel because they all originate as copies from that single corrupted ballot. So, it does not make a difference in terms of *ballot reliance*. It could, however, make a difference in terms of improving *communication reliance*.²⁵

In terms of trust, the above example motivates including not only multiple communication channels but also channels of diverse types, with machine-machine communication (e.g., transmission channels) and human-machine communication (e.g., ballot channels). Human-human communication (e.g., audit channels) should also be considered because we want machines to be verifiable independently of the machines themselves. We further want the voter to be able to act as a source and as verifier in more than one part of the system, in both human-human and human-machine communication.

What is needed is thus to include redundant and diverse²⁶ communication channels at each node, which can be used to provide correction channels. For example, at different layers and corresponding to different time, space, reference frame, source, recipient, verifier, context, and environment conditions, even during the election and in real time.

In other words, there should be sufficient diversity in implementing the desired redundancy in order to enable different verifiers to qualify reliance on information (i.e., trust ¹⁵) in ways that are not just self-referential (subjective).

As trust conditions, the VITM requires redundancy and diversity:

Table 3. Trust Conditions

<i>Redundancy</i>	Use multiple channels and types of communication.
<i>Diversity</i>	Channels must be at least partially independent.

For example, in enabling multiple and diverse verification channels but without breaking the secret ballot condition, a voting system could allow a voter to verify whether her ballot is present or not at the ballot box, whether her ballot at the ballot box is valid, and (with an acceptable degree of confidence) whether the ballot is what she saw and cast. It is important to note that random verification

²⁵ According to Section 6.3, the ballot box would more probably receive the right ballot (even though corrupted) for $N = 500$ than for $N = 1$. With more transmission channels, there is less probability that a majority of channels can be perturbed at the same time.

²⁶ Diversity does not require absolute independence. If two channels are not 100% mutually dependent, the probability that both may fail at the same time is smaller than that of any single one to fail.

by even a small fraction of the voters (e.g., 5%)²⁷ can be effective in detecting errors and create a credible deterrent to fraud. Thus, when properly done, voter verification by even a small fraction of the voters can help improve election outcome trustworthiness for all voters.

Another characteristic of a good voting system is freedom of choice. A voter needs to be able to trust that she cannot be threatened, hurt, or even denied something as a result of her vote choices. Election operators and the public in general also need to be able to trust that voters cannot receive favors as a result of their vote choices. To assure freedom of choice, the only person to whom the vote should be proved is the voter himself. In other words, no one else and certainly not the election operators should be able to prove how a voter voted. Otherwise, the vote could be coerced or sold. Thus, no information channels may exist allowing individual voters and ballots to be linkable. We call this the *unlinkability* condition; even though voters and votes must each be identifiable, they must not be linkable to each other.

In terms of electronic versus paper ballot voting systems, the primary concern for increasing reliability is the capacity of the correction channels compared with the capacity of the error channels —not the physical properties of the medium (e.g., paper) used in a communication channel. If an electronic voting system is able to provide N proofs (human and machine based), these N proofs for some value of N larger than one will become more reliable than one so-called “physical proof” even if this one proof is engraved in gold or printed on paper. The make-up of each channel’s carrier (e.g., paper, photons, electrons) is by itself irrelevant. See [40] for further discussion on this topic.

To assure end-to-end trust, in addition to protect casting the ballot, one must also protect the former steps in presenting the ballot as well as the latter steps in tallying and auditing the ballot. This will be given as a set of Requirements (Section 7) that work together in an end-to-end design. The concept of trust in [15] has the same meaning in all the components [VITM, Requirements, WVS] of the design. Different verifiers can also use different trust models (e.g., hierarchical, web-of-trust), which are all integrated (though not unified) through the same trust definition, subsuming the *physical* and *conceptual* cases.

With potential applications to other security problems, we note that the oft-cited security paradigm “the weakest link defines the security of the system” does not apply here. We also do not rely on “perfect” parts or one “strong” evidence. The WVS error-free design condition (see Section 6.3, Error-Free Condition) is based on several, mostly independent (and possibly imperfect) evidences that can build a correction channel with enough capacity so as to correct all but an arbitrarily small fraction of the errors. This is a new security paradigm provided by this approach [16, 17, 40, 41], which we call the mesh paradigm—a mesh does not break if a link breaks.

With the WVS, one or more witnesses are used to capture the primary information: what the voter sees and confirms on the screen. A primary information witness may be used by itself as the voted ballot, possibly with better reliability

²⁷ A conservative estimate obtained by applying the Saltman auditing model [43].

than the voted ballot kept solely under EO control. However, one “strong” evidence can never be perfectly strong —it may, and will, fail. The objective of the WVS is thus *not* to rely on one “strong” evidence, which can never be perfectly strong, but to rely on several, mostly independent evidences.

Instead of saying “all parts are perfect” or “there really has to be security on every single piece”, which is impossible to obtain as one piece will inevitably be the weakest link, we say “there really has to be one or more alternate secure paths in case any single piece fails, because fail it may”.

Instead of a “Fort Knox” approach (“make it stronger”) that relies on what becomes a single point of failure (or congestion), this approach calls for a mesh of links such that a number of links may fail at the same time without compromising accuracy, reliability, and voter privacy.

The same solution applies to preventing faults and fraud, but we start with a “Default Denial” policy that also originates from trust considerations —trust is earned [15]. In other words, everything is denied until acceptable proof that it should not be. And acceptable proof must come in more than one way, and must be verified in more than one way, as qualified reliance on information [15] (see Table 3, Trust Conditions).

6.5 Preferred Setup

The VITM is based on our Information Transfer Model (ITM) [16], which uses the conceptual separation of a *subject* into witness-objects (observable entities, as witnesses or references for chosen properties of the *subject*) and reader-objects (observer entities, as adequate readers of the witnesses).

To implement the VITM in a preferred setup that can be directly implemented as a Witness-Voting System (WVS, Section 8), we analyze the voting process and define a first *subject* property to be the election outcome. During the election, witness-objects will *witness* events and then become available as observable entities for reader-objects at specific time periods, including for tallying and auditing.

Next, we define the witness-objects (hereafter, witnesses) and reader-objects (hereafter, readers) that the verifiers need to establish. Further consideration is provided in Section 8.3. Independently of any witnesses and readers set up by a particular verifier called the Election Operator (EO), the VITM allows witnesses and readers to be added at any step of the process by other verifiers. The witnesses and readers shall be designed to be privacy-preserving, so that there is less limitation who the verifiers may be or how they are supposed to act.

If verifiers wish to add more witnesses (readers) than what may be desirable in terms of a practical design, a *cut-and-choose* strategy can be applied to allow a smaller number of witnesses (readers) to be chosen without bias. To reduce complexity, the *end-to-end argument*²⁸ can be used to preferably place witnesses, for example, at the start point A (what the voter sees and casts) and the end point B (the tally results).

²⁸ Instead of demanding complete and correct control at every intermediate step, control the end points [44].

7 Voting System Requirements

The Voting System Requirements (Requirements) are limitations to prevent conceptual and physical interference. Mirroring the broad scope of interference as defined in this work (section 6.2), the Requirements will include functional and performance aspects as well as environmental and non-functional aspects. With such a comprehensive approach we want the Requirements to be expressive enough to comprise a variety of means that can be falsely used to influence elections without voting, including interference that does not even exist physically and just stays as a perceived threat.

The Requirements are derived using the VITM considerations in Section 6 and work together with the VITM to setup a conforming voting means (the WVS, Section 8). Some requirements were already naturally motivated in the VITM, such as the secret ballot, one ballot per voter, and transparency. Further consideration is provided in [40], extended in this work.

The VITM does not require open source software as a *sine qua non* condition since, as Linux demonstrates, even a long development time and thousands of eyes do not guarantee accuracy and reliability. Bugs, fraud, virus, Trojan horses and faults may still influence the outcome, without possibility of detection [49] even with open source software. The VITM solution to the software (and hardware) reliability question is further discussed in Section 8.3.

As the final and simple step in calculating election outcome, tallying can use open source to allay error concerns. Following the *optimal design*, detection and correction of errors is provided when diverse tallying modules are used and the outputs compared. To prevent fraud, tallying modules should consider information on a strict “need to know” basis. Tallying should not receive any information that it does not need, and it should not produce any information that is not needed to define the election outcome. For example, the voter’s ethnicity or choice of ballot language should not be a consideration in tallying. An important requirement, thus, is that the cast ballot should be only choice-dependent, so that it must be independent from all other types of data (e.g., the cast ballot must be representation-independent and language-independent).

7.1 Privacy Considerations

Voter privacy is necessary to prevent coercion and vote buying. It is also, often, a legal requirement (see footnote 1).

The voter privacy condition is at times confused with anonymity. However, to assure election integrity voters must not be anonymous. Both the voter and the vote must be and are well-known at different stages of the election process. Yet, because no one should be able to link votes with voters (*unlinkability*), if we know the voter (e.g., in voter registration) we cannot know the vote that was cast by that voter; if we know a vote (e.g., in tallying) that was cast, we cannot know the voter who cast it. All voters are identified and still the election results are anonymous.

Often in elections, the available choices need to be defined and controlled per voter and per group of voters, which raises privacy concerns. For example, when using different ballot styles with different choices due to jurisdiction, geopolitical or other differences; when ballot rotation is used to assure fairness of option placement in a list of options (different voters see the same options but with a different top to bottom sequence); when law requires (as in some US states to assure non-discrimination compliance) that each voter’s ethnicity must be registered; or when allowing more than one language or media (e.g., audio, Braille printing, large fonts).

To prevent coercion and vote buying, the choice of voting method is also significant. For example, while postal mail voting cannot prevent voter coercion, precinct-based voting creates a protected environment where voter coercion may be prevented. Online voting, even if not precinct-based, may also prevent coercion [18].

However, even when voting is private and the voting method allows coercion to be preventable, the voter privacy requirement may have further and subtle consequences. For example, the ballots cast should not be disclosed to anyone (not even during or after tallying); just the tallied results can be disclosed. The reason is that choice patterns that are likely to be statistically unique in an election, and yet include a desired outcome, can be defined and then used as a “*voter pattern fingerprinting*” mechanism to identify a single voter, or all voters of a small group (e.g., one family), which may influence an election by means of coercion and vote buying²⁹ Although without identifying voters, disclosing the cast ballots can also be passively used to influence next elections, as a detailed glimpse into voter demographics that can later be used with *gerrymandering* (footnote 20) and yet finer methods such as social pressure. Conversely, if the cast ballots are disclosable (e.g., to the election operators) then they should be made public to all, so that all stakeholders could equally benefit from their analysis.

To clearly define the concept of voter privacy, we previously [40] discerned not only different types of voter privacy but also different “strengths”. The list below presents this classification, and additional comments, ranked from lowest to highest privacy strength.

Policy privacy: Exemplified by election systems that depend on election officials and/or separated machines in order to protect voter privacy. Policy privacy cannot prevent the operators or attackers from penetrating both systems and rejoining the information. It also cannot prevent a court order that would mandate rejoining the information in the servers.

Computational privacy: Exemplified by election systems that rely upon a *quorum* of verifiers or election operators, in blind signatures, mix-servers or homomorphic encryption, such that not less than N people working together (which defines a *threshold of collusion*) can compromise voter privacy. Such systems rely not only on absence of design flaws, but also on absence of a compromise to the computational platform (e.g., a virus that would record all N keys for later use, a

²⁹ This method has been used by organized crime (private communication).

bug, or a non-intrusive electromagnetic eavesdropping device that would record all N keys for latter reuse without ever physically penetrating the platform). To work, this method also depends on non-physical assumptions, including that the election officials shall use independent judgment, cannot be coerced or intimidated, are not bound by a conflicting trust commitment such as of a military, political or religious nature, do not constitute a cabal, and have a minimum level of honesty to resist collusion. It also assumes that the control system that enforces the threshold of collusion cannot be corrupted. The *quorum* method depends on a number of assumptions that in general are not revealed to the voters and are likely not suitable for public elections. This practice also fails to protect voter privacy under court or administrative order (when all keys and secrets must be revealed).

Information-theoretic privacy: Exemplified by election systems in which there is no reliance on cryptography in order to protect privacy (e.g., no reliance on public-key encryption). It defines a privacy strength that cannot be broken by computation, even with unbounded time and resources. Information-theoretic privacy, however, fails in the following examples: (a) parties share keys in advance and use one-time pads, which is impractical and subject to collusion (when keys are revealed); (b) parties share physically protected channels, which fails against collusion where the channel is compromised (also without detection); (c) parties share information (via secret-sharing techniques) and they are assumed not to pool it together, which fails against collusion. Information-theoretic privacy also cannot protect voter privacy in the case of a court order that mandates revealing all keys and secrets used in the system.

Fail-safe privacy: Defined in [40] for election systems where voter privacy cannot be compromised even if everything fails including software and hardware, everyone colludes and there is a court order that mandates revealing all keys and secrets used in the system. Current paper ballot voting systems can provide fail-safe voter privacy.³⁰

7.2 Summary of Requirements

These Requirements are an extension of our previous work [40] and apply to voting systems and rules of any type. In terms of systems architecture, our goal is that the Requirements present a comprehensive consideration as to *what* is to be done (functional), *how well* a voting system is to perform (performance) and under *what conditions* it is to operate (environmental and non-functional). *Requirements are created and dictated by the goal of minimizing interference.*

³⁰ Fingerprints and DNA may be left on paper ballots by voters. If not prevented (e.g., by using a selection mask), this could be used to compromise privacy. However, the cost and resources in mounting such an analysis has been a deterrent in practice. Another way to compromise privacy is by matching paper fibers alongside a tear-off boundary, for paper ballots that provide a “receipt” to voters; this would, however, require the cooperation of the voter.

1. Fail-safe voter privacy. The inability to link voters with votes is required. Voter privacy **MUST** be assured even if everything fails to function properly, or is forced to function improperly, everyone colludes and there is a court order to reveal all election data, without time limitation.

2. Collusion-free vote secrecy. The inability to know individual votes is required. Vote secrecy **MUST** be assured even if all election means (e.g., voted ballots) and security keys are made known by an attack or a fault (i.e., vote secrecy **MUST NOT** depend only on communication protocol and cryptographic assumptions, or on a threshold of collusion for the key holders).

3. Verifiable election integrity. The inability of any number of parties to influence the outcome of an election except by properly voting is required. The system **MUST** provide verifiability that each vote tallied originated from an eligible voter and that all votes are tallied as seen and cast by voters. For any voter the system **MUST** also provide verifiability that there is one and only one valid ballot cast by the voter in the ballot box.

4. Fail-safe privacy in verification. Voters **MUST NOT** have to disclose their identity in order to verify their votes or report a perceived error. Fail-safe voter privacy (Requirement #1) **MUST** be preserved even when voters participate in a verification process.

5. Physical recounting and auditing. **MUST** provide for reliability in auditing and vote recounting, with an error rate as low as desired. The auditing and vote proofs **MUST** be capable of being physically stored offline and verified for integrity in real-time during the election, without compromising any other Requirement and allowing effective human verification.

6. 100% accuracy. Each vote or absence of vote (blank vote) **MUST** be correctly counted, with accuracy (spread of a single measurement) error as close to zero as desired. Counting and recounting ballots **MUST NOT** reduce accuracy.

7. Manifold of links (mesh system). **MUST** use a manifold (mesh) of redundant links and keys to securely define, authenticate and control ballots. **MUST** avoid single points of failure or congestion—even if improbable.

8. Offline secure control structure. **MUST** provide an offline secure end-to-end control structure for presenting and collecting information from voters (e.g., ballots). **MAY** use digital certificates under a single issuing authority. The control **MUST** be data-, representation-, and language-independent.

9. Authenticated choice representation. The representations of the choices available to each voter, including ballot style and ballot rotation if ballots are used, **MUST** be authenticated and **MUST** be provided with a control means that also authenticates the voter.

10. Authenticated user-defined presentation. If voters **MAY** choose language, font size, layout, display format, and other presentation properties, the choices **MUST** be authenticated but **SHOULD NOT** be provided with a control means that also authenticates the voter.

11. Allow voter to review and change choices before casting ballot. **MUST** allow voters to review and change choices from “vote” to “blank vote” or

to any other available choice, at will, for any race and for any number of times, before casting their votes. This is equivalent to receiving a new blank ballot.

12. Allow undervote (abstain). SHOULD allow voters to abstain to vote in any or all choices (undervote). An undervote MAY be represented by a specific choice, such as Abstain or No Vote. The voter MAY receive a warning of undervoting. However, such a warning SHOULD NOT be public.

13. Overvote warning. To prevent mistakes and post-voting fraud: if overvoting is detected, SHOULD warn the voter that a vote has to be cleared in order to proceed. Any warning SHOULD be made known only to the voter, without public disclosure. MAY prohibit overvoting.

14. Provide for null ballots. MAY allow voters to null races or even the entire ballot as an option (e.g., to counter coercion; to protest against lack of voting options). Overvoting MAY be used as a mechanism to provide for null ballots.

15. Technology independent. SHOULD not depend on any specific technology to support any Requirement.

16. Open review, open code. SHOULD allow all source code to be publicly known and verified (open source code, open peer review).

8 A Conforming Voting Means: The WVS

The Witness-Voting System (WVS) is a physical device that follows the *optimal design* of section 6.3 and the Requirements, both physically and conceptually.

The significant aspect of the *optimal design* is the Error-Free Condition $C \geq E$ (hereafter, EFC). As a non-limiting example, Table 4 shows an implementation class of the WVS where the EFC is divided in two parts.³¹

Table 4. WVS Implementation Using a Four-Step Program

I	EFC 1: Optimal casting and counting of votes, in order to
II	assure voters that their ballot choices are private and
III	assure election outcome trustworthiness.
IV	EFC 2: If (II) and (III) are satisfied, SUCCESS; else improve (I).

A first EFC is used to optimize *accuracy and reliability* in the casting and counting of votes, and operates primarily within the first and second boundaries (see boundary definitions in Section 6). A second EFC is used to optimize *voter privacy and election outcome trustworthiness*, where operational feedback at step (IV) is used to provide physical (e.g., witness configuration) and conceptual (e.g., Requirements) correction channels by the election operators, the stakeholders, the voters, and other elements within the third boundary.

The following features result by applying Table 4 to Sections 2 and 8.1.

³¹ The *optimal design* does not specify how the EFC should be implemented; for example, one could use an open-loop, a closed-loop with feedback, or a mixed approach.

The WVS design is transparent. As part of the election setup, the WVS design invites stakeholders to add their own witnesses and readers, which is critical to the trustworthiness of the election’s outcome, step (III).

The WVS design is simple, scalable, and auditable. The WVS design uses multiple, diverse channels to transfer information from each voter to the tally results, which is promotive of end-to-end auditing and is amenable to end-to-end arguments (see section 6.5). Consequently, the number of process points that need to be witnessed is relatively small and the design can be implemented using parallel elements (e.g., witnesses, readers, tally processing) for simplicity, scalability and reliable auditing.

The WVS design is robust. For example, in order to satisfy step (III) the WVS implementation should be transparent in operation to the voters and the operators. If that does not seem to be the case for some operation in (I), that operation can be redesigned using the considerations in Section 6 and the problem solved to the satisfaction of both (II) and (III).

The WVS design is extensible. Additional control variables can be added. For example, a step (IIIa) could be inserted to promote low election cost, while a step (IIIb) could measure voter feedback to assure accessibility compliance.

8.1 How It Works

With the objective of highlighting the basic concepts used in our approach, section 2 presented a WVS implementation that was highly simplified and used intuitive requirements. A more practical WVS implementation is presented here, following the model presentation and qualified requirements of the previous sections. Additional WVS implementations are described in [14].

This presentation includes many considerations of our approach. Notably missing from our implementations, but referenced in [17], we did not provide examples of voter registration, voter authentication, and ballot authentication, as well as their use in terms of specifying a “closed-circle” voting process.

Most importantly, the WVS captures what we call the “magic moment”, when the voter sees and confirms the choices in order to cast the ballot. This is the *primary information* that needs to be voter-verified and universally verifiable, albeit anonymously and without creating a “voter pattern fingerprinting” vulnerability (Section 7.1). The WVS will also provide end-to-end verification that ballots are processed correctly, giving voters and other parties independent verification capability that the votes are cast, collected, and counted as intended.

Without limitation, we consider that the message selected by a voter (i.e., the ballot that a voter sees and casts) is transmitted initially to the ballot box, stored there and later tallied, with totals made public. We further describe how the influence of errors in the voting process (including fraud, malfunctions, passive and active attacks) may be corrected to achieve an outcome error that can be as close to zero as desired (*error-free*), using ways of transmitting and receiving the information that are provably optimal in reducing errors.

We will often refer the reader to slides in [14] for visualization. The slides are available online. Please load or print slide number 13 in [14]. To refer to

elements in the slide we will use their three-digit number tags, such as (230) for the witness device.

Slide 13 in [14] exemplifies a conventional electronic voting machine (also called DRE, Direct Recording Electronic voting machine) with a first implementation of the witness device (230) within dotted lines, containing the beam-splitter module³² (212), camera (216), computer (220) and cartridge B.

A voter (200) provides input voting data through an input device (202) which provides an interface to a first computer (204). The computer (204) comprises an electronic voting machine such as a DRE and includes an optical projector (206) for projecting voter selection data on a touch screen or other display device (208) which displays a ballot and receives a voter's selections. The touch screen (208) provides an image of the voter's selections to the voter (200).

The image on the touch-screen (208) is created from a projector (206) that transmits a light beam along path (210) to a beam-splitter module (212). Part of the light continues along a path to the screen (208). Light deflected by the beam splitter module (212) is provided to a digital camera such as a charge coupled device CCD (216). Device (216) makes an image record of pixel data representing the voter's selections. The image record is communicatively coupled by data link or lead (218) to computer (220). Computer (220) thus provides an unparsed record of pixel data corresponding to the image on the screen (200) at the time the voter confirms the ballot. Computer (220) may compress, authenticate and encrypt the pixel image data, but does not parse or otherwise interpret the optical image captured by element (216).

The "magic moment" occurs when the voter confirms the ballot selections by accepting the data through an accept/reject interface (224); the accept interface actuates the computer (220) to store the pixel data corresponding to a voter's ballot selections in a storage device, Cartridge B (ballot box B) at (222). Data are stored in random fashion in Cartridge B to safeguard voter privacy. One method for enabling the accept interface to actuate computer (220) to store the pixel data representative of the voter's confirmed selections could be for the voter's "accept" action, such as the click of a mouse over a predefined image area, a hyperlink color change or a button changing color, or a screen border change or other readily discernible video actuator, to intrinsically trigger an activation event such as which the module (216) would record and thereby trigger the computer (220) to store the pixel data at the time of the activation event. That is, computer (220) would be responsive to a video event activated by the voter which would indicate confirmation of the voter's selections. If the voter rejects the selections, the video event is not activated and the image data are not stored. The voter, who following a procedure defined by an election official may notify and be authorized by an election official, may repeat the voting process.

Witness record capture can also be used for audio events or for any other type of output or assistive device, as in the case of voters who are visually impaired, or other data, such as for example, the data used to create and record a voter's

³² A beam splitter is a simple optical device that splits a beam of light in two. An ordinary piece of glass can split a beam of light in two beams.

selections in Braille, as may be used to communicate the voters selections to the voter, for verification and authentication.

The witness device (230) makes an image record of the data evidencing the voting event, such as the pixel data or other verifiable data record of the portion of computer memory used for generating the ballot and corresponding selections for each voter. Thus, the witness provides an efficient representation of voting selections without parsing or interpreting the data. The witness module (230) can be further seen as a means for taking a snapshot or image of pixel data or other literal data in a selected portion of computer memory as such data would represent, at the time of confirmation, the ballot and selections made by each voter. An important aspect of the witness (230) is when it does not interpret or effect any change in the voting data, also when any transformations in the memory data recorded by the witness are independent of content. The witness is applied in an example to generate an event record of the ballot and voter's selections, as evidence of the act of voting. Witness (230) thereby provides transparency to the voter while ensuring reliability and accuracy of outcome as intended by the voter (230). The witness further protects the system against virus or attacks intended to change the data.

The witness device (230) approaches an ideal communication channel, with a very simple code that can be verified to have zero loss, no viruses and no bugs; however, absence of errors is not required (see Sections 8.3 and 2). The witness (230) can thus make a video image, audio image or other snapshot of the specific portion of computer memory used to generate the voting interface and the voter's selections at the point of authentication by the voter. Witness (230) also can be adapted with a means for sending a verification of the voter's selections back to the voter (200) via a communication link and printer for providing the voter with a voting receipt directly from the witness, that the voter can see but not touch; the voting receipt should then be cut, for random storage purposes, and dropped in a paper ballot box (not shown in the diagram).

The data corresponding to the authenticated vote by voter (200) is stored in cartridge B and sent to the results module (232), which could be located at a precinct. The memory image data from computer (220) may also be stored in cartridge (B) to provide independent verification of the voting result. The witness may be open to public scrutiny without having any effect on the data.

Computer (204), such as a DRE voting system, also records the voters selections from input device (202) and upon authentication by the voter, stores the selections in cartridge A and sends a signal representative of the voter's selections to the result (232). The results from cartridge A and from cartridge B are stored for tallying at the results module (232), which is a reader object that may be multiply provided by the EO and stakeholders.

The different witnesses and readers provide a means for auditing the accuracy and reliability of various input data streams. In the event of a discrepancy, the results are collected by a difference resolution module (234) and provided for comparison against records of votes stored in cartridge B, representative of the one or more cartridges attached to the witness.

The outcome or election result (236) is calculated according to the election rules, which also define the recount procedures (238), if any. The recount procedures (238) may use a different number of witnesses from cartridge A and of stored votes from cartridge B.

As another possibility, the computer (220) in witness (230) stores only the digital signature of the image data in cartridge B. That is, at the time the voter (200) authenticates the vote selections, computer (220) is activated, as set forth above, to record and store in cartridge B only the digital signature of the image, without the image. If necessary, as in the case of a discrepancy, the data from cartridge A can be used to recreate the image corresponding to its digital signature in cartridge B in accordance with techniques which are well known.

Storage of digital signature data has significant advantages over storing ballot image data. The size of compressed, encrypted ballot image data is on the order of 500 kilobytes or more, for a single ballot. In contrast, digital signature data size is on the order of 500 bytes. Thus, digital signature data can require three orders of magnitude less storage space than compressed image data. For an application such as electronic voting where large amounts of election data must be transmitted and stored, storage of digital signature data may provide an efficient, compact witness for the image data transmitted from cartridge A to the result (232). Storage of digital signature data may also provide advantages in terms of increased processing speed and makes more efficient use of memory and system resources in a typical electronic voting environment.

The use of an optical witness device is not mandatory. Other slides of [14] show how similar arrangements (in different WVS implementations) can be made for an *electrical witnesses* of the electric signal that provides the image, for a *display card witness* in the DRE that holds the pixels shown in the screen, instead of or in addition to one or more *optical witnesses*.

A significant aspect of this design is that witnesses and readers from each interested election stakeholder can be used at the same time and place. For example, witnesses representing political parties A, B and C could be provided by each party to anonymously watch each ballot *as seen and cast by the voter*. Diverse tally modules, as reader objects, can also be provided by parties A, B and C, where the tally process can be executed in parallel and employ diverse error-correcting algorithms.

Slide number 16 in [14] shows how the inputs from various witness and readers can be combined and evaluated, including error-detection, by weighed consensus to calculate the election outcome at the final tally. Weighed consensus is a well-known technique to increase fault-tolerance in the presence of interference and is used here as a non-limiting error-correcting algorithm example to combat faults and fraud (modeled as interference).

While the WVS may use closed-source components (e.g., EO's or stakeholders' witness elements may include proprietary code), open-source elements can also be used and their outputs openly compared for mutual verification at every step of the process, including at the final tally, to assure election outcome trustworthiness.

8.2 Perspective

Here we provide a road-map to understand the WVS development.

Voting systems comprise four main components: [17]

- (i) a registration service for verifying and registering legitimate voters,
- (ii) voting stations where the voter makes choices on a ballot,
- (iii) a device called the ballot box where the ballot is collected, and
- (iv) a tallying service that counts the votes and announces the results.

This work focuses on the latter three parts, but it can also be applied to improve the trustworthiness of voter registration.³³

As it should be clear, this work does not propose any change in voting *per se* but a qualitative change in how we understand and mathematically model voting. Our approach comes from the realization that even though (a) voting is deterministic, (b) does not include random sampling, and (c) all ballots must be counted, it innately uses communication processes where, when we look close enough, information must be defined by a stochastic model as pointed out in Information Theory [9, 11].

The principles behind our approach are well-known since 1948 and have stood the test of time as standard practice in modeling and optimizing information flow, reliability, and availability. We note that contrary to its common use in cryptography [10], Information Theory was not used in this work with the goal to provide or improve upon communication secrecy *per se*, even though it is used for example in meeting the *secret ballot* requirement.

In this work we used the Voting Information Transfer Model (VITM) to model, in terms of information transfer, how voting works and how the election outcome can be reliably and accurately measured, from the moment the voter asks for a ballot to the moment that all votes are tallied and made public.

Our approach to voting has been publicly discussed since January 2000, with public input [45, 46] and with the help from experts who worked on and verified elections in the US and abroad for more than 25 years.

Leading a public discussion [47] at the Brookings Institute, we emphasized early on the need for *unlinkability* (see Section 7.1) regarding fail-safe privacy assurances in voting. This is a basic condition in our approach. *The significant aspect is that in order to preserve election integrity, the only person to whom the vote is proved should be the voter himself.*

Also in 2000, we presented some early results of the VITM application with the Multi-Party protocol [41], which has been in continuous use by Safevote with online voting in elections worldwide.

In 2001 we proposed and solved the “*fundamental problem in voting*” [18] also using the VITM. Work on voting systems and requirements [18, 40] provided the

³³ Voter registration receives little scrutiny in general, yet it can create numerous difficulties and disenfranchise many voters. Efforts to manipulate the number and political affiliation of persons voting may compromise an election even if the voting system is *error-free*.

foundation for presenting the Witness-Voting System (WVS) later in 2001 [14], now extended in this work.

The components [VITM, Requirements, WVS], which provide the framework used here, are based on our 1997 extensions of Information Theory: (1) we include interference caused by faults as well as attacks and threats (adversaries) [16] in the concept of noise; (2) we add the concept of trust [15]; and (3) we define an Information Transfer Model (ITM) [16]. Extensions (1) and (2) are commented in Sections 6.2 and 6.4, while extension (3) is used in Section 6. The ITM uses (1) and (2) to achieve measurements with an error as small as desired in the presence of fault, security and threat considerations. The ITM has been in continuous development and, recently, the ITM was applied to qualitatively improve privacy and security in email communications [48].

8.3 Witnesses and Readers

The WVS uses the VITM, which is a model based on observables (i.e., witnesses or references) and observers (i.e., adequate readers of the witnesses).

Witnesses and readers may be “public” (i.e., independently accessible) or restricted to a set of parties (e.g., within a qualified security boundary). For transparency, often witnesses should be public, meaning that multiple parties (possibly also adversaries) are able to access them. The WVS design is open to the inclusion of public witnesses and readers, which more easily invites stakeholders to be part of the election setup and assures transparency regarding any step that may be seen as critical to the trustworthiness of the election’s outcome. Further consideration is provided in [16].

A reader allows the information contained in the witnesses to be properly used by a verifier; thus witnesses and readers must be “adequate”. However, perfect functionality or full independence are *not* required in order for witnesses and readers to be useful in reducing the effects of errors and fraud (see Sections 6.3 and 6.4).

An important question is what can we trust if both the software and the hardware cannot be trusted? It is well-known that software cannot be trusted [49]. The same applies to hardware, where counterfeit or malicious components³⁴ can compromise the very platform where an otherwise trusted software runs.

Attacks such as defined in [49] and [50] are included in the definition of *interference* used in this work and presented in Section 6.2. Accordingly, the redundancy and diversity in the VITM/WVS design can increase *reliability* and combat perturbations caused by such attacks. The number of different components and implementations that we need to use, and how diverse they may be, is set and adjusted operationally by the Error-Free Condition (Section 6.3) and correction channel considerations (Section 6.4). As noted in Section 6.4, the oft-cited security paradigm “*the weakest link defines the security of the system*” does not apply here.

³⁴ This threat is not new and defenses already exist (e.g., the MIL-SPEC process, Orange Book). However, as shown in the work of King *et. al.* [50], such an attack is becoming easier to create and more difficult to detect.

9 Conclusions

Our presentation scope focuses on voting. We preface our conclusions, however, by noting that this approach can also be applied to improve the trustworthiness of voter registration and other aspects of an election, such as ballot design and provisional voting. Voter registration receives little scrutiny in general, yet it can create numerous difficulties and disenfranchise many voters. Efforts to manipulate the number and political affiliation of persons voting may compromise an election even if the voting system is error-free. The Requirements presented here can also be used to guide the development of more effective legal regulations.

This work shows that although voting is a deterministic process, the long-standing problem of election outcome trustworthiness cannot be described, much less solved, deterministically. Information is essentially stochastic in its nature and voting, as a process that transfers information from voters to tally results, is no exception in its use of information.

The *Voting Information Transfer Model* (VITM) presented in this work is an information transfer model based on observables (i.e., witnesses or references) and observers (i.e., adequate readers of the witnesses). The VITM applies to any type of voting, with or without ballots, paper based, electronic, or online.

The VITM allows us to look deeper into the information flow of the voting process, where *information has to be modeled stochastically* using Information Theory. However, to prevent any confusion, we emphasized that *the ballot seen and cast by a voter is not a stochastic variable in our approach*.

Anything that perturbs the election outcome is defined as *interference*, the same definition used in Information Theory. The VITM further distinguishes interference with functional and performance influence (called *physical interference*) from interference with environmental and non-functional influence (called *conceptual interference*). Accordingly, our formalism is expressive enough to comprise a variety of means that can be falsely used to influence elections without voting, including interference that does not even exist physically but merely stays as a perceived threat.

The VITM directly uses results previously developed in 60 years of experience with Information Theory to define a provably *optimal design* that can reduce election outcome error to a value as close to zero as desired, which we call *error-free*, establish comprehensive Voting System Requirements (Requirements) to combat interference, and implement a conforming voting means—the Witness-Voting System (WVS).

The [VITM, Requirements, WVS] define the three components of our framework, describing a realistic voting system environment that includes interference of the election outcome from faults, attacks and threats by adversaries.

In such a framework, requirements are not arbitrary. *Requirements are created and dictated by the goal of minimizing interference*. Some Requirements are naturally motivated in the VITM, such as the secret ballot, one valid ballot per voter, and transparency. We present sixteen Requirements, including functional, performance, environmental and non-functional considerations, presenting a comprehensive consideration as to *what* is to be done (functional), *how*

well a voting system is to perform (performance) and under *what conditions* it is to operate (environmental and non-functional). The Requirements meet or exceed current public-election voting requirements. With our approach, the voter and the vote are unlinkable (secret ballot) although each is identifiable.

We showed that conceptual interference can be physically prevented by the WVS. Conversely, ignoring such possibility in the system design could create conditions for unintended conceptual and physical interference.

The WVS can achieve an error-free election outcome by optimally preempting, or at least resolving, any dispute regarding *accuracy, reliability, voter privacy, and election outcome trustworthiness*.

The WVS design is open to the inclusion of public witnesses and readers, including diverse cast ballot witnesses and error-correcting modules for tallying. This invites stakeholders to be part of the election setup and assures transparency regarding any step that may be seen as critical to the trustworthiness of the election's outcome.

We showed, quite generally and with potential applications to other security problems, that the oft-cited security paradigm “the weakest link defines the security of the system” does not apply to the WVS. *Perfection of each human and each element of hardware and software is not required*. Perfect independence is also not required in order for witnesses and readers to be useful in reducing the effects of errors and fraud. Rather, a central aspect of the WVS is that, exactly because we know that *all elements are imperfect*, there are enough (as qualified in Section 6.3, Error-Free Condition) *multiple correction channels* providing feedback in order to enable the WVS to *fully* offset the influence of interference such as caused by faults, attacks and threats by adversaries.

We discussed two WVS conforming implementations. The first WVS was highly simplified and used intuitive requirements, with the objective of highlighting the basic concepts used in our approach. A more practical WVS implementation followed the model presentation and the qualified requirements, including many considerations of our approach. Notably missing from our implementation discussions here, but referenced in our approach, we did not provide examples of voter registration, voter authentication, and ballot authentication, as well as their use in terms of specifying a “closed-circle” voting process.

The Witness-Voting System can also be applied to existing electronic voting machines (e.g., DRE), including “black box” voting machines (with closed-source software), to verify their accuracy and reliability before, after, and during an election. Paper based voting systems, including optical scan ballots, may also benefit by using the Witness-Voting System as a verification enhancement and in providing multiple correction channels.

One of the most used correction channels is still the human factor (e.g., reading the paper ballot). Yet, this does not mean that the human factor *must* remain as the single correction channel in voting. We consider all-electronic and online voting systems as firm possibilities when the capacity of correction channels is increased.

Mounting economic, political and social factors press for an evolution in voting, a par with everything else. More voter convenience with less cost and less

time to vote, higher usability, higher public confidence in the election process, reduced voter coercion in online voting (compared to mail voting), robust voter registration, voter-verified auditing, increased opportunities for voter participation, reduced costs in the management of elections, secure storage of ballots, and reduced time for tabulation and auditing are among the practical results that can be quantified with the framework described in this paper. These and other beneficial results have been confirmed in the continuous use of this framework since 2000 with online voting in elections worldwide.

Acknowledgments. This work was supported by Safevote, Inc., which retains intellectual property rights and copyright. We are thankful for comments and references from many collaborators and several online discussion groups. Comments by L. Bernstein, T. Gerck, V. Neppe, M. Norden, A. Todd and an anonymous reviewer were particularly helpful in this paper.

References

1. Gritzalis, D. (ed.): Secure Electronic Voting. Kluwer Academic Publishers, Holland (2002) ISBN 1-4020-7301-1
2. Bannet, J., Price, D.W., Rudys, A., Singer, J., Wallach, D.S.: Hack-a-Vote: Demonstrating Security Issues with Electronic Voting Systems. IEEE Security Privacy Magazine 2(1), 32–37 (2004)
3. Jones, D.W.: Evaluation of voting technologies. In: Secure Electronic Voting, ch. 1, *ibid*
4. Circuit Court of the Second Judicial Circuit, Leon County, Judge William Gary, CASE NO. 2006-CA-2973 (2006), Copy online at, <http://electionlawblog.org/archives/ess-pdf.pdf>
5. Gumbel, A.: Steal This Vote: Dirty Elections and the Rotten History of Democracy in America. Nation Books (2005)
6. Gerck, E.: Year 2000 Public Sector U.S. Market Intelligence Study. The Bell, vol. 1(1-3) (2000) ISSN 1530-048X, Copy online at <http://thebell.net/archives/thebell1.1.pdf>, <http://thebell.net/archives/thebell1.2.pdf>, <http://thebell.net/archives/thebell1.3.pdf>
7. Gerck, E.: Editorial –The Difference Between the Right and the Almost-Right. The Bell 1(7) (2000) ISSN 1530-048X, Copy online at <http://thebell.net/archives/thebell1.7.pdf>
8. Cherry, S.: The Perils of Polling. IEEE Spectrum Online (2004), <http://www.spectrum.ieee.org/oct04/4036>
9. Shannon, C.E.: A Mathematical Theory of Communication. Bell System Technical Journal 27, 623–656 (1948), Copy online at <http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf>
10. Shannon, C.E.: Communication Theory of Secrecy Systems. Bell System Technical Journal 28, 656–715 (1949), Copy online at, <http://netlab.cs.ucla.edu/wiki/files/shannon1949.pdf>
11. Gallagher, R.G.: Information Theory and Reliable Communication. John Wiley & Sons, New York (1968)
12. Yu, F.T.S.: Optics and Information Theory. John Wiley & Sons, New York (1976)

13. Cover, T.M., Thomas, J.A.: Elements of Information Theory. John Wiley & Sons, New York (1991)
14. Gerck, E.: Voting With Witness Voting. In: WOTE 2001 Seminar, Tomales Bay, Calif, Caltech-MIT Voting Technology Project (2001), Copy online at <http://www.vote.caltech.edu/wote01/pdfs/gerck-witness.pdf> and <http://safevote.com/doc/gerck-witness.pdf>
15. Feghhi, J., Feghhi, J., Williams, P.: Digital Certificates: Applied Internet Security. In: Gerck, E. (ed.) Trust Points, pp. 194–195. Addison-Wesley, Reading (1998) ISBN 0-20-130980-7, Copy and additional material online at <http://mcwg.org/mcg-mirror/trustdef.htm>, <http://nma.com/papers/it-trust-part1.pdf>
16. Gerck, E.: Certification: Extrinsic, Intrinsic and Combined. Published online by the MCG (1997), Copy online at, <http://mcwg.org/mcg-mirror/cie.htm>
17. Gerck, E.: Private, secure and auditable Internet voting. In: Secure Electronic Voting, ch. 11, *ibid*
18. Gerck, E.: Voting Systems: from Art To Science. In: Caltech MIT Voting Technology Conference, Pasadena, Calif. (2001), Copy online at http://www.hss.caltech.edu/~voting/gerck_present.ppt, http://safevote.com/doc/gerck_present.ppt
19. Saltman, R.G.: Independent Verification: Essential Action to Assure Integrity in the Voting Process. NIST Report, Order No. SB134106W0703 (2006)
20. Saltman, R.G.: Effective Use of Computing Technology in Vote-Tallying, Report NBSIR 75-687 (republished as NBS Special Publication 500-30, 1978), National Institute of Standards and Technology, Gaithersburg, MD (1975)
21. Chaum, D.: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. Communications of the ACM 24(2), 84–88 (1981)
22. Cohen, J., Fischer, M.: A robust and verifiable cryptographically secure election scheme. In: Proc. 26th IEEE Symposium on Foundations of Computer Science (FOCS '85), pp. 372–382. IEEE Computer Society, Los Alamitos (1985)
23. Benaloh, J.: Verifiable Secret-Ballot Elections. PhD thesis, Yale University, Dept. of Computer Science (1987)
24. Mercuri, R.T.: Physical Verifiability of Computer Systems. In: Proc. of the 5th International Computer Virus and Security Conference (March 1992)
25. Cramer, R., Franklin, M., Schoenmakers, B., Yung, M.: Multi-authority secret ballot elections with linear work. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 72–83. Springer, Heidelberg (1996)
26. Benaloh, Tuinstra: Receipt-free Secret-ballot Elections. In: STOC '94, pp. 544–553 (1994)
27. Gerck, E.: Contra Costa County Shadow Election Report. Final Project Report to the California Secretary of State (2000), Copy online at <http://safevote.com/doc/SafevoteContraCostaCountyElection.pdf>, <http://safevote.com/onlineballot.htm>
28. Neff, C.A.: A Verifiable Secret Shuffle and its Application to E-Voting. In: Proceedings ACM-CCS 2001, pp. 116–125 (2001)
29. Jakobsson, M., Juels, A., Rivest, R.L.: Making Mix Nets Robust for Electronic Voting by Randomized Partial Checking. In: Proc. Usenix Security, pp. 339–353 (2002) IACR reprint 2002/025
30. Kiayias, A., Yung, M.: Robust Verifiable Non-Interactive Zero-Sharing: A Voting Utility For Enhanced Privacy. In: Secure Electronic Voting, ch. 9, *ibid*
31. Mercuri, R.T., Neumann, P.G.: Verification of Electronic Balloting Systems. In: Secure Electronic Voting, ch. 3, *ibid*

32. Chaum, D.: Secret-Ballot Receipts: True Voter-Verifiable Elections. *IEEE Security & Privacy* 2(1), 38–47 (2004)
33. Chaum, D., Ryan, P.Y., Schneider, S.A.: A Practical, Voter-verifiable, Election Scheme, Tech. Report Series CS-TR-880, School of Computer Science, Univ. of Newcastle upon Tyne (2004)
34. Chaum, D., Carback, R., Sherman, A., Clark, J., Popoveniuc, S., Vora, P.L.: Scantegrity: End-to-End Voter-Verifiable Optical-Scan Voting. *IEEE Security & Privacy* 6(3), 40–46 (2008)
35. Garera, S., Rubin, A.D.: An Independent Audit Framework for Software Dependent Voting Systems. In: 14th ACM Conference on Computer and Communications Security (November 2007)
36. Wysong, T.: Open Source and Open Software. *The Bell* 1(5) (2000) ISSN 1530-048X, Copy online at <http://thebell.net/archives/thebell1.5.pdf>
37. Kitcat, J.: Why Electronic Voting Software Should Be Free Software. *The Bell* 1(5) (2000) ISSN 1530-048X, Copy online at <http://thebell.net/archives/thebell1.5.pdf>
38. Caltech/MIT Voting Technology Project Report: What Is, What Could Be (July 2001), Copy online at <http://vote.caltech.edu>
39. Bowen, D.: Dr. Strangevote or: How I Learned to Stop Worrying and Love the Paper Ballot. In: *The Security Symposium Keynote, USENIX Security* (2008)
40. Gerck, E.: Voting System Requirements. In: 5th International Conference Proceedings Financial Cryptography. LNCS, vol. 2339, pp. 243–268. Springer, Heidelberg (2002), An earlier version is available online at <http://www.thebell.net/papers/vote-req.pdf>
41. Gerck, E.: From Voting to Internet Voting. *The Bell* 1(1), 5 (2000) ISSN 1530-048X, Copy online at <http://thebell.net/archives/thebell1.1.pdf>
42. Mason, D.M.: Rethinking Political Geography. *The Bell* 1(8), 5 (2000) ISSN 1530-048X, Copy online at <http://thebell.net/archives/thebell1.8.pdf>
43. Saltman, R.G.: The Strength of Small Numbers. *The Bell* 1(6), 5 (2000) ISSN 1530-048X, Copy online at <http://thebell.net/archives/thebell1.6.pdf>
44. Saltzer, J.H., Reed, D.P., Clark, D.D.: End-to-end arguments in system design. *ACM Transactions on Computer Systems (TOCS)* 2(4), 277–288 (1984)
45. *The Bell Newsletter* ISSN 1530-048X, Archive online at <http://thebell.net/archives/>
46. IVTA Tech WG online archive <http://www.mail-archive.com/techivta.org/>
47. Gerck, E.: Public Comments. In: “The Future of Internet Voting”, Brookings Institute Symposium, Washington D.C (January 2000), Copy online at <http://www.brookings.edu/events/2000/0120elections.aspx>
48. Gerck, E.: Secure Email Technologies X.509/PKI, PGP, IBE and Zmail. In: Krishna, S.J., Raju, E. (eds.) *Corporate Email Management*, ch. 12, pp. 171–196. ICFAI University Press (2007) ISBN 81-314-12797, Copy online at <http://email-security.net/papers/pki-pgp-ibe-zmail.pdf>
49. Thompson, K.: Reflections on Trusting Trust. *Communications of the ACM* 27(8), 761–763 (1984)
50. King, S.T., Tucek, J., Cozzie, A., Grier, C., Jiang, W., Zhou, Y.: Designing and Implementing Malicious Hardware. In: *Proceedings of the First USENIX Workshop on Large-Scale Exploits and Emergent Threats LEET* (April 2008), Copy online at http://www.cs.uiuc.edu/homes/kingst/Research_files/king08.pdf

Coercion-Resistant Electronic Elections

Ari Juels¹, Dario Catalano², and Markus Jakobsson³

¹ RSA Laboratories

Bedford, MA, USA

ajuels@rsa.com

² Università di Catania

Catania, Italy

catalano@dmi.unict.it

³ PARC

Palo Alto, CA, USA

mjakobsson@parc.com

Abstract. We introduce a model for electronic election schemes that involves a more powerful adversary than previous work. In particular, we allow the adversary to demand of coerced voters that they vote in a particular manner, abstain from voting, or even disclose their secret keys. We define a scheme to be *coercion-resistant* if it is infeasible for the adversary to determine whether a coerced voter complies with the demands.

A first contribution of this paper is to describe and characterize this newly strengthened adversary. In doing so, we additionally present what we believe to be the first formal security definitions for electronic elections of *any* type. A second contribution is a protocol that is provably secure against our formalized adversary. While strong attack models are of theoretical interest, we emphasize that our results lie close to practicality in two senses: We model real-life threats (such as vote-buying), and our proposed protocol combines a fair degree of efficiency with low structural complexity. While previous schemes have required an untappable channel, ours has the more practical requirement of an anonymous channel.

Keywords: coercion-resistance, electronic voting, mix networks, receipt-freeness.

1 Introduction

Many voters participating in shareholder elections in the United States regularly cast ballots over the Internet [1]. Geneva, Switzerland adopted Internet voting for civic elections in 2004 (enhanced by quantum cryptography in 2007) [45]. Similarly, tens of thousands of members of the UMP political party in France participated by Internet in a national Presidential primary in 2007 [46] [1].

These are just a few instances of a broadening trend toward Internet-based voting. While voting of this kind appears to encourage higher voter turnout [44] and make accurate accounting for votes easier, it also brings with it a heightened risk of large-scale error and manipulation. A number of papers in the cryptographic literature have thus

¹ The integrity of the vote in this case was specially ensured by the availability of only one candidate.

described ways of achieving robust and verifiable electronic elections, in which ballots and processing data are posted to a publicly accessible bulletin board. For some examples (but not by any means an exhaustive list), see [8,15,18,22,23,28,32,38,42,47,50]. This literature is distinct from the extensive body of work on the security of Direct-Recording Electronic (DRE) machines, freestanding tallying devices in common use today in public polling places. Researchers have demonstrated serious, fundamental vulnerabilities in widely fielded DRE systems, e.g., [33].

There are two other threats, however, that it is equally crucial to address in a fair and democratic election process: We speak of *voter coercion* and *vote buying*. Internet-based voting does not introduce these problems, but it does have the potential to exacerbate them by extending the reach and data collection abilities of an attacker. This has been highlighted in one way by a notorious (possibly satirical) Web site that supported the auctioning of votes [2]. Seller compliance was in that case merely voluntary. Conventional Internet voting schemes, however, including those described in the literature, actually provide an attacker with ready-made tools for verifying voter behavior and thereby exerting influence or control over voters. Without careful system design, the threats of coercion and vote buying are potentially far more problematic in Internet voting schemes than in ordinary, physical voting schemes.

One commonly proposed way of achieving secure electronic voting systems is to use a cryptographic system known as a *mix network* [14]. This is a tool that enables a collection of servers to take as input a collection of ciphertexts and to output the corresponding plaintexts according to a secret permutation. A straightforward way to achieve an election system that preserves the privacy of voters, then, is to assign a private digital signing key to each voter. To cast a ballot, the voter encrypts her choice and signs it, and then posts it to a bulletin board (i.e., a publicly accessible memory space). When all ballots have been collected and the corresponding signatures have been checked, the ciphertexts are passed through a mix network. The resulting plaintext versions of the voter choices may then be tallied. Thanks to the privacy preserving property of the mix network, an adversary cannot tell which vote was cast by which voter. This approach is frequently advocated in the mix-network literature, as in, e.g., [8,14,23,28].

In an ordinary mix-based scheme of this kind, an adversary can coerce a voter straightforwardly. The adversary can simply furnish the voter with a ciphertext on a particular candidate, and then verify that the voter posted a ballot containing that ciphertext. Alternatively, the adversary can demand the private signing key of the voter and verify its correctness against the corresponding public key. An adversary attempting to buy votes can use the same means. Other types of cryptographic voting schemes, namely homomorphic schemes [5,18] and schemes based on blind signatures [21,42], suffer from similar vulnerabilities.

1.1 Previous Work

Previous investigations of coercion-resistant voting have been confined to a property known as *receipt-freeness*. Roughly stated, receipt-freeness is the inability of a voter to prove to an attacker that she voted in a particular manner, even if the voter wishes to do so. For a more formal definition, see [42]. The property of receipt-freeness ensures that

an attacker cannot determine exact voter behavior and therefore cannot coerce a voter by dictating her choice of candidate. It also protects against vote-buying by preventing a potential vote buyer from obtaining proof of the behavior of voters; voters can thereby *pretend* to sell their votes, but defraud the vote buyer. The notion of receipt-freeness first appeared in work by Benaloh and Tuinstra [5]; their scheme, based on homomorphic encryption, was shown in [26] not to possess receipt-freeness as postulated. An independent introduction of the idea appeared in Niemi and Renvall [40]. Okamoto [41] proposed a voting scheme which he himself later showed to lack the postulated receipt-freeness; a repaired version by the same author, making use of blind signatures, appears in [42]. Sako and Kilian [48] proposed a multi-authority scheme employing a mix network to conceal candidate choices, and a homomorphic encryption scheme for production of the final tally. The modelling of their scheme was clarified and refined by Michels and Horster [36]. The Sako and Kilian scheme served as a conceptual basis for the later work of Hirt and Sako [26], followed by the more efficient approach of [3]; these two are the most efficient (and correct) receipt-free voting schemes to date. A recently proposed scheme by Magkos *et al.* [35] distinguishes itself by an approach relying on tamper-resistant hardware, but is flawed.²

All of these receipt-free voting schemes include somewhat impractical assumptions. For example, these schemes assume the availability of an *untappable channel* between the voter and the authorities, that is, a channel that provides perfect secrecy in an information-theoretic sense. (I.e., even encryption does not provide an untappable channel.) The scheme in [42] makes the even stronger assumption of an *anonymous* untappable channel. (It is also not very practical in that it requires voter interaction with the system three times in the course of an election.) Moreover, all of these schemes (excepting [42]) lose the property of coercion-resistance if the attacker is able to corrupt even one of the tallying authorities in a distributed setting. The scheme of Hirt and Sako still retains coercion-resistance when such corruption takes place, but only under the strong assumption that the voter knows *which* tallying authorities have been corrupted; the proposal of Baudron *et al.* has a similar property.

In a systems-level analysis confined to the special case of DREs, Karlof, Sastry, and Wagner [31] have identified vulnerabilities in the influential and innovative schemes of Chaum and Neff [20,39]. In particular, covert-channels in these schemes open up the possibility of various forms of coercion. Also in context of such DRE systems, Moran and Naor [37] have formalized and shown how to achieve the property of receipt-freeness and proposed a partial solution to the problems identified by Karlof *et al.*.

Apart from their often impractical assumptions, there is a more serious problem with all of the receipt-free voting schemes described in the literature. *Receipt-freeness*

² We are unaware of any other mention of a break of this scheme in the literature, and therefore briefly describe one here. The Magkos *et al.* system employs an interactive honest-verifier ZK proof made by a smartcard to the voter. Presumably because of the simulability of this proof, the authors describe the proof as being “non-transferable.” This is not quite true, however. In particular, an adversary can stipulate that the voter engage in the proof using a challenge that the adversary has pre-selected. The proof then becomes transferable, yielding a means of receipt construction by the adversary. As noted in [26], this type of attack also explains why *deniable encryption* [13] does not solve the problem of coercion in a voting system.

alone fails to protect against several forms of serious, real-world attack in election systems, among them:

Randomization attack: This attack was noted by Schoenmakers in 2000 [51]; he described its applicability to the scheme of Hirt and Sako. The idea is for an attacker to coerce a voter by requiring that she submit randomly composed balloting material. In this attack, the attacker (and perhaps even the voter) is unable to learn what candidate the voter cast a ballot for. The effect of the attack, however, is to nullify the choice of the voter with a large probability. For example, an attacker favoring the Republican party in a United States election would benefit from mounting a randomization attack against voters in a heavily Democratic district.

Forced-abstention attack: This is an attack related to the previous one based on randomization. In this case, the attacker coerces a voter by demanding that she refrain from voting. All of the schemes cited above are vulnerable to this simple attack. This is because the schemes authenticate voters directly in order to demonstrate that they are authorized to participate in the election. Thus, an attacker can see who has voted, and use this information to threaten and effectively bar voters from participation.³

Simulation attack: The receipt-free schemes described above assume that the attacker cannot coerce a voter by causing her to divulge her private keying material after the registration process but prior to the election process. Such an attack, however, is a real and viable one in previously proposed schemes, because these permit an attacker to verify the correctness of private keying material. For example, in [42], the voter provides a digital signature which, if correct, results in the authority furnishing a blind digital signature. In [26], the voter, when casting a ballot, proves knowledge of a private key relative to a publicly committed or published value. In general, receipt-freeness does not prevent an attacker from coercing voters into divulging private keys or buying private keys from voters and then *simulating* these voters at will, i.e., voting on their behalf.

1.2 Our Contribution

We make a twofold contribution in this paper, which is an extended version of work appearing in [30]. First, we investigate a stronger and broader notion of coercive attacks than receipt-freeness. This notion, which we refer to as *coercion-resistance*, captures what we believe to be the fullest possible range of adversarial behavior in a real-world, Internet-based voting scheme. A coercion-resistant scheme offers not only receipt-freeness, but also defense against randomization, forced-abstention, and simulation attacks—all potentially in the face of corruption of a minority of tallying authorities. We propose a formal definition of coercion-freeness in this paper. Two other properties are essential for any voting scheme, whether or not it is coercion-resistant. These are *correctness* and *verifiability*. As formal definitions for these properties are to

³ An exception is the scheme in [42], which does not appear to be vulnerable to a forced-abstention attack. This is because the scheme seems to assume that the authority checks voter enrollment privately. In other words, the scheme does not permit public verification that participating voters are present on a published voter roll. This is potentially a problem in its own right.

the best of our knowledge lacking in the literature, we provide those definitions as well. We thus provide what we believe to be the first formal security framework for electronic elections in general.

To demonstrate the practical realizability of our definitions, we describe a voting scheme that possesses the strong property of coercion-resistance proposed in this paper—and also naturally possesses the properties of correctness and verifiability. Our scheme does not require untappable channels, but instead assumes voter access to an anonymous channel at some point during the voting process. Anonymous channels can be realized in a practical way by use of mixnets, e.g., [23,38], while untappable channels require largely unrealistic physical assumptions. We note that anonymous channels are in fact a minimal requirement for *any* coercion-resistant schemes: An attacker that can identify which voters have participated can obviously mount a forced-abstention attack. A drawback of our scheme is that, even with use of asymptotically efficient mix networks as in [23,38], the overhead for tallying authorities is quadratic in the number of voters. Thus the scheme is only practical for small elections. Our hope and belief, however, is that our proposed scheme might serve as the basis for refinements with a higher degree of practical application.

1.3 Intuition Behind Our Scheme

In a conventional voting scheme, and also in receipt-free schemes like [26], the voter V_i identifies herself at the time she casts her ballot. This may be accomplished by means of a digital signature on the ballot, or by an interactive authentication protocol. The key idea behind our scheme is for the identity of a voter to remain hidden during the election process, and for the validity of ballots instead to be checked blindly against a voter roll. When casting a ballot, a voter incorporates a concealed credential. This takes the form of a ciphertext on a secret value σ that is unique to the voter. The secret σ is a kind of *anonymous credential*, quite similar in spirit to, e.g., [9,10]. To ensure that ballots are cast by legitimate voters, the tallying authority \mathcal{T} performs a blind comparison between hidden credentials and a list L of encrypted credentials published by an election registrar \mathcal{R} alongside the plaintext names of registered voters.

By means of mixing and blind comparison of ciphertext values, it is possible to check whether a concealed credential is in the list L or not, without revealing which voter the credential has been assigned to. In consequence, an attacker who is given a fake credential $\tilde{\sigma}$ by a coerced voter cannot tell whether or not the credential is valid. (The attacker will learn how many ballots were posted with bad credentials. Provided, however, that some spurious ones are injected by honest players, authorities, or even outsiders, the individuals associated with bad ballots will remain concealed.) Moreover, the attacker cannot mount randomization or forced-abstention attacks, since there is no feasible way to determine whether an individual voter has posted a ballot or not. In particular, after divulging fake credential $\tilde{\sigma}$, a voter can go and vote again using her real credential σ .

1.4 Organization

In section 2, we describe our setup and attack models and sketch a few of the major adversarial strategies. We provide formal definitions in section 3. We describe the

particulars of our proposed scheme in section 4, prefaced by a summary of the underlying cryptographic building blocks. We give a detailed outline for proof of the coercion-resistance of our scheme in section 5 and conclude in section 6.

2 Modelling

An election system consists of several sets of entities:

1. *Registrars*: Denoted by $\mathcal{R} = \{R_1, R_2, \dots, R_{n_R}\}$, this is a set of n_R entities responsible for jointly issuing keying material, i.e., credentials to voters.
2. *Authorities (Talliers)*: Denoted by $\mathcal{T} = \{T_1, T_2, \dots, T_{n_T}\}$, authorities are responsible for processing ballots and jointly counting votes and publishing a final tally.
3. *Voters*: The set of n_V voters, denoted by $\mathcal{V} = \{V_1, V_2, \dots, V_{n_V}\}$, are the entities participating in a given election administered by \mathcal{R} . We let i be a public identifier for V_i .

We make use of a *bulletin board*, denoted by \mathcal{BB} . This is a piece of universally accessible memory to which all players have appendive-write access. In other words, any player can write data to \mathcal{BB} , but cannot overwrite or erase existing data. Moreover, voters will be able to read the contents of \mathcal{BB} once the vote casting phase has ended. For notational convenience, we assume that data are written to \mathcal{BB} in μ -bit blocks for an appropriate choice of μ . Shorter data segments may be padded appropriately. For simplicity of exposition, we assume no ordering on the contents of \mathcal{BB} .

2.1 Functions

We define a *candidate slate* \mathcal{C} as an ordered set of n_C distinct values $\{c_1, c_2, \dots, c_{n_C}\}$, each of which corresponds to a voter choice, typically a candidate or party name. In an election, choice c_j may be identified according to its index j . Thus, for cryptographic purposes the candidate slate consists of the integers $\{1, 2, \dots, n_C\}$ and may be specified by n_C alone. We define a *tally* on an election under slate \mathcal{C} to be a vector \mathbf{X} of n_C positive integers x_1, x_2, \dots, x_{n_C} such that x_j indicates the number of votes cast for choice c_j . The protocols composing an election system are then as follows:

- **Registering**: The function $\text{register}(SK_{\mathcal{R}}, i, k_1) \rightarrow (sk_i, pk_i)$ takes as input the private registrar key $SK_{\mathcal{R}}$, a (voter) identifier i and a security parameter k_1 , and outputs a key pair (sk_i, pk_i) . This is computed jointly by players in \mathcal{R} , possibly in interaction with voter V_i .
- **Voting**: The function $\text{vote}(sk, PK_{\mathcal{T}}, n_C, \beta, k_2) \rightarrow \text{ballot}$ takes as input a private voting key, the public key of the authorities \mathcal{T} , the candidate-slate specification n_C , a candidate selection β , and a security parameter k_2 , and yields a ballot of bit length at most μ . The form of the ballot will vary depending on the design of the election system, but is in essence a digitally signed vote choice encrypted under $PK_{\mathcal{T}}$.
- **Tallying**: The function $\text{tally}(SK_{\mathcal{T}}, \mathcal{BB}, n_C, \{pk_i\}_{i=1}^{n_V}, k_3) \rightarrow (\mathbf{X}, P)$ takes as input the private key of the authority \mathcal{T} , the full contents of the bulletin board, the candidate-slate size, all public voting keys, and a security parameter k_3 and outputs a vote tally \mathbf{X} , along with a non-interactive proof P that the tally was correctly computed.

- **Verifying:** The function $\text{verify}(PK_{\mathcal{T}}, \mathcal{BB}, n_C, \mathbf{X}, P) \rightarrow \{0, 1\}$ takes as input the public key of the authorities, the contents of the bulletin board, the candidate-slate size, the voting tally, and a non-interactive proof of correct tallying. It outputs a ‘0’ if the tally is incorrect and a ‘1’ otherwise.

We define an election scheme ES as the collection of these functions. Thus $\text{ES} = \{\text{register}, \text{vote}, \text{tally}, \text{verify}\}$.

Remark: There are many election models in use throughout the world. The model we propose here excludes important variants. In some systems, for example, voters are asked to rank candidate choices, rather than just listing those they favor. Many systems permit the use of *write-in* votes, i.e., the casting of a ballot in favor of a candidate not listed on the slate for the election. We exclude write-in voting from our model because it undermines the possibility of coercion resistance in any scheme where an observer can see a complete election tally including write-in votes. An attacker may, for example, require coerced voters to cast write-in ballots for candidate names consisting of random strings pre-specified by the attacker. This way, the attacker can: (1) Verify that coerced voters complied with instructions, by looking for the random strings the attacker furnished, and (2) Ensure that the votes of coerced voters are not counted, since random strings will most likely not correspond to real election choices. (Thus, this would combine the forced abstention attack and the randomization attack.)

2.2 Summary of the Attack Model

We consider the process for a single election as proceeding in these phases, corresponding largely with the functions enumerated in section [2.1](#):

1. **Setup:** If not already available, key pairs are generated for or by \mathcal{R} and \mathcal{T} . The candidate slate C for the election is published by \mathcal{R} with appropriate integrity protection.
2. **Registration:** The identities and eligibility of would-be participants in the election are verified by \mathcal{R} . With successful verification, an individual becomes a registered voter, receiving from \mathcal{R} a credential permitting participation in the election. Previously registered voters can re-use their credentials. \mathcal{R} publishes voter roll L .
3. **Voting:** Referring to the candidate slate C , registered voters use their credentials to cast ballots.
4. **Tallying:** The authority \mathcal{T} processes the contents of the bulletin board \mathcal{BB} so as to produce a tally vector \mathbf{X} specifying the outcome of the election, along with a proof of correctness P of the tally.
5. **Verification:** Any player, whether or not an election participant, can view \mathcal{BB} , P and L to verify that the tally produced by \mathcal{T} in the previous phase is correct.

Assumptions in setup phase: Our security definitions allow static, active corruption by the adversary of a minority of players in \mathcal{R} and \mathcal{T} in the setup phase. The security of our construction then relies on generation of the key pairs $(SK_{\mathcal{T}}, PK_{\mathcal{T}})$ and $(SK_{\mathcal{R}}, PK_{\mathcal{R}})$ by a trusted third party, or, alternatively, on an interactive, computationally secure key-generation protocol such as [\[25\]](#) between the players in \mathcal{R} and those in \mathcal{T} .

Assumptions prior to registration: The adversary may coerce a voter prior to the registration phase by requiring that the voter retain transcripts of the registration process, or by trying to dictate the voter’s future interaction with the registrar.

Assumptions in registration phase: We make the assumption that the registration phase proceeds without any corruption of voters. This assumption is at some level a requirement for a coercion-free election, as an attacker capable of corrupting and seizing the credentials of a voter in this initial phase can mount a simulation attack. More precisely, we must make *at least one* of three assumptions about the registration phase:

1. Erasure of data from voter interaction with \mathcal{R} is compulsory by the voter (e.g., enforced by smartcards provided to voters). This prevents an attacker from requesting registration transcript data after the fact; or
2. The adversary cannot corrupt any players in \mathcal{R} ; or
3. Voters become aware of the identity of any corrupted player in \mathcal{R} .

The reason we require at least one of these assumptions is as follows. If none of these assumptions holds, then the adversary can, on demanding information from a voter, verify the correctness of some portion thereof, where the voter would not know what portion is being checked. In other words, the adversary can perform spot checks, with a high probability of successfully detecting false transcripts. In consequence, the adversary can coerce voters into divulging full transcripts of their interactions with \mathcal{R} , thereby enabling a simulation attack. In contrast, if at least one of the assumptions holds, we show that it is possible to formulate a protocol that is coercion-resistant.

Assumptions on voting, tallying and verification phases: After registration, we assume that the adversary can seize control of a minority of players in \mathcal{T} and any number of voters in a static, active manner. (Since \mathcal{R} does not participate after registration, we need not consider adversarial corruption of \mathcal{R} at this point.) The adversary may also attempt to coerce voters outside its control by having them divulge private keying material⁴ or behave in a prescribed manner in voting. Voters are assumed to cast their ballots via fully anonymous channels, i.e., channels such that an attacker cannot determine whether or not a given voter cast a ballot. This assumption is a requirement for any election scheme to be fully coercion-resistant: If an attacker can tell whether or not a given voter cast a ballot, then the attacker can mount a forced-abstention attack. In practice, an anonymous channel may be achieved by letting voters cast ballots in busy public places, by use of anonymizing, asynchronous mix-networks, etc.

3 Formal Definitions

We now turn our attention to formal security definitions of the essential properties of *correctness*, *verifiability*, and *coercion-resistance*, respectively abbreviated *corr*, *ver*, and *c-resist*. Our definitions hinge on a set of experiments involving an adversary \mathcal{A}

⁴ We assume that coercion takes place remotely. That is, the adversary may not watch over the shoulder of a voter, monitor her hard-drive, etc. Our proposed protocol does defend against some shoulder-surfing, however, by permitting voters to use fake keys and/or re-vote.

in interaction with components of the election system ES. This adversary is assumed to retain state throughout the duration of an experiment. We formulate our experiments such that in all cases, the aim of the adversary is to cause an output value of ‘1’. Thus, for experiment $\text{Exp}_{\text{ES},\mathcal{A}}^E(\cdot)$ on property $E \in (\text{ver}, \text{corr}, \text{c-resist})$, we define $\text{Succ}_{\text{ES},\mathcal{A}}^E(\cdot) = \Pr[\text{Exp}_{\text{ES},\mathcal{A}}^E(\cdot) = \text{‘1’}]$.

According to the standard definition, we say that a quantity $f(k)$ is *negligible* in k if for every positive integer c there is some l_c such that $f(k) < k^{-c}$ for $k > l_c$. In most cases, we use the term negligible alone to mean negligible with respect to the full set of relevant security parameters. Similarly, in saying that an algorithm has *polynomial running time*, we mean that its running time is asymptotically bounded by some polynomial in the relevant security parameters. As the properties of correctness and verifiability are general ones that apply to any voting scheme, we discuss them first. We then consider coercion resistance, the special focus of our work here.

Correctness: We first consider the property of correctness. It is a twofold property: First, it stipulates that an adversary \mathcal{A} cannot pre-empt, alter, or cancel the votes of honest voters, i.e., those not *controlled*. Second, it stipulates that \mathcal{A} cannot cause voters to cast ballots resulting in double voting, i.e., use of one credential to vote multiple times, where more than one vote per credential is counted in the tally.

For a strong definition of correctness, we give the adversary (artificially) strong powers. Apart from getting to select a set V of voters she will control, she can choose the candidate-slate size n_C , and choose what votes will be cast by voters she does not control. If the adversary still cannot cause an incorrect tally to be computed (i.e., one not corresponding to the votes cast), then the scheme has the correctness property. The adversary aims to cause more than $|V|$ ballots to be counted in the final tally on behalf of the controlled voters, or to alter or delete the vote of at least one honest voter. (This means that: (1) The verification of the tally succeeds, and (2) That either a vote is “dropped” or “added.”) Our definition assumes implicitly that tally is computed correctly by the authority \mathcal{T} . (The next property we consider, verifiability, addresses the possibility that this is not so.) We let $\langle Y \rangle$ here denote the multiset on to entries in the vector Y , and $|Y|$ denote the cardinality of set Y .

Experiment $\text{Exp}_{\text{ES},\mathcal{A}}^{\text{corr}}(k_1, k_2, k_3, n_C, n_V)$

$\{(sk_i, pk_i) \leftarrow \text{register}(SK_{\mathcal{R}}, i, k_2)\}_{i=1}^{n_V};$	% voters are registered
$V \leftarrow \mathcal{A}(\{pk_i\}_{i=1}^{n_V}, \text{“choose controlled voter set”});$	% \mathcal{A} corrupts voters
$\{\beta_i\}_{i \notin V} \leftarrow \mathcal{A}(\text{“choose votes for uncontrolled voters”});$	% \mathcal{A} votes for honest voters
$\mathcal{BB} \leftarrow \{\text{vote}(sk_i, PK_{\mathcal{T}}, n_C, \beta_i, k_2)\}_{i \notin V};$	% honest voters cast ballots
$(X, P) \leftarrow \text{tally}(SK_{\mathcal{T}}, \mathcal{BB}, n_C, \{pk_i\}_{i=1}^{n_V}, k_3);$	% honest ballots are tallied
$\mathcal{BB} \leftarrow \mathcal{A}(\text{“cast ballots”}, \mathcal{BB});$	% \mathcal{A} posts ballots to \mathcal{BB}
$(X', P') \leftarrow \text{tally}(SK_{\mathcal{T}}, \mathcal{BB}, n_C, \{pk_i\}_{i=1}^{n_V}, k_3);$	% all ballots are tallied
if $\text{verify}(PK_{\mathcal{T}}, \mathcal{BB}, n_C, X', P') = \text{‘1’}$ and	% does verify accept?
$(\{\beta_i\} \not\subset \langle X' \rangle$ or $ \langle X' \rangle - \langle X \rangle > V)$ then	% did \mathcal{A} 's tampering work?
output ‘1’;	
else	
output ‘0’;	

We say that ES possesses the property of correctness if for all polynomial-time adversaries \mathcal{A} , it is the case that $\text{Succ}_{\text{ES}, \mathcal{A}}^{\text{corr}}(k_1, k_2, k_3, n_V)$ is negligible.

Verifiability: As explained above, an election system has the property of correctness if computation of tally always yields a valid tabulation of ballots. Given the ability of an adversary \mathcal{A} , however, to corrupt some number of authorities among \mathcal{T} , we cannot be assured that tally is always computed correctly. The property of verifiability is the ability for any player to check whether the tally \mathbf{X} has been correctly computed, that is, to detect any misbehavior by \mathcal{T} in applying the function tally.

A strong security definition for verifiability is appropriate given the high level of auditability required for trustworthy elections. Such a definition considers an attacker \mathcal{A} capable of controlling *all* of the voters and tallying authorities in \mathcal{T} . This attacker seeks to construct a set of ballots on \mathcal{BB} and a corresponding tally \mathbf{X} and proof P of correct tabulation such that the proof is accepted by verify, but the tally is in fact incorrect. By an incorrect tally, we mean one in which all of the valid ballots of a particular voter (i.e., corresponding to a particular credential) are discounted, or else where multiple votes are tallied that could have been generated by the same voting credential. Our experiment characterizing verifiability is as follows.

```

Experiment  $\text{Exp}_{\text{ES}, \mathcal{A}}^{\text{ver}}(k_1, k_2, k_3, n_C, n_V)$ 
   $\{(sk_i, pk_i) \leftarrow \text{register}(SK_{\mathcal{R}}, i, k_2)\}_{i=1}^{n_V};$            % voters are registered
   $(\mathcal{BB}, \mathbf{X}, P) \leftarrow$                                      %  $\mathcal{A}$  concocts full election
     $(\mathcal{A}(SK_{\mathcal{T}}, \{(sk_i, pk_i)\}_{i=1}^{n_V}, \text{"forge election"});$  %  $\mathcal{A}$  concocts full election
   $(\mathbf{X}', P') \leftarrow \text{tally}(SK_{\mathcal{T}}, \mathcal{BB}, n_C, \{pk_i\}_{i=1}^{n_V}, k_3);$  % tally is taken on  $\mathcal{BB}$ 
  if  $\mathbf{X} \neq \mathbf{X}'$                                              %  $\mathcal{A}$ 's tally = correct  $\mathcal{BB}$  tally?
    and verify( $PK_{\mathcal{T}}, \mathcal{BB}, n_C, \mathbf{X}, P$ ) = '1' then      % does function verify accept?
      output '1';
  else
    output '0';

```

We say that ES possesses the property of verifiability if for all positive integers n_V and all adversaries \mathcal{A} with polynomial running time, the quantity $\text{Succ}_{\text{ES}, \mathcal{A}}^{\text{ver}}(k_1, k_2, k_3, n_V)$ is negligible. A technical strengthening of this definition and that for correctness is possible, and discussed in the next section, appendix [A](#), of this paper.

Another aspect of verifiability that we do not formally define, but do mention here and incorporate into our proposed protocol is that of verification against voter rolls. In particular, it may be desirable for any election observer to check that credentials were assigned only to voters whose names are on a published roll. This is not technically a requirement if we rule out corruption of players \mathcal{R} , but may still be desirable for high assurance of election integrity. Our definitions can be modified accordingly.

Coercion resistance: Coercion resistance may be regarded as an extension of the basic property of privacy. Privacy in an election system is defined in terms of an adversary

that cannot interact with voters during the election process. In particular, we say that an election is private if such an adversary cannot guess the vote of any voter better than an adversarial algorithm whose only input is the election tally. (Note, for example, in an election where all voters vote Republican, the system may have the property of privacy, even though the adversary knows how all voters cast their ballots in that election.)

Coercion resistance is a strong form of privacy in which it is assumed that the adversary may interact with voters. In particular, the adversary may instruct targeted voters to divulge their private keys subsequent to registration, or may specify that these voters cast ballots of a particular form. If the adversary can determine whether or not voters behaved as instructed, then the adversary is capable of blackmail or otherwise exercising undue influence over the election process. Hence a coercion-resistant voting system is one in which the user can deceive the adversary into thinking that she has behaved as instructed, when the voter has in fact cast a ballot according to her own intentions.

Our definition of coercion resistance requires addition of a new function to voting system ES:

- The function $\text{fakekey}(PK_{\mathcal{T}}, sk, pk) \rightarrow \tilde{sk}$ inputs the public key of the authorities and the private/public key pair of the voter. It outputs a spurious key \tilde{sk} .

Of course, for the function fakekey to enable coercion resistance, the key \tilde{sk} must be indistinguishable by the adversary \mathcal{A} from a valid key, and only distinguishable by a majority of talliers \mathcal{T} . This property is captured in our experiment characterizing coercion resistance. To simplify the formulation of the experiment, we assume implicitly that tally is computed by an oracle (with knowledge of $SK_{\mathcal{T}}$). It suffices, however, for \mathcal{T} to be computed via a protocol that achieves correct output and is computationally simulable by the adversary \mathcal{A} (who, it will be recalled, may corrupt a minority of \mathcal{T}).

Our definition of coercion resistance centers on a kind of game between the adversary \mathcal{A} and a voter targeted by the adversary for coercive attack. A coin is flipped; the outcome is represented by a bit b . If $b = 0$, then the voter casts a ballot with a particular choice β , and provides the adversary with a false voting key \tilde{sk} ; in other words, the voter attempts to evade adversarial coercion. If $b = 1$, on the other hand, then the voter submits to the coercion of the adversary; she simply furnishes the adversary with her valid voting key sk , and does not cast a ballot. The task of the adversary is to guess the value of the coin b , that is, to determine whether or not the targeted voter in fact cast a ballot. We permit the adversary in this definitional game to specify the ballot value β . While it is somewhat unnatural for the adversary thus to specify the intention of the voter, this permits us to achieve the strongest possible security definition.

If the adversary has perfect knowledge about the intentions of all voters, then coercion is unavoidable. For example, if the adversary is attempting to coerce one voter in a given election and knows that all hundred of the other eligible voters will cast ballots,

then the adversary can mount an abstention attack straightforwardly. The adversary in this case simply threatens the voter in the case that the total tally for the election is one hundred and one. Similarly, suppose that the adversary does not know whether or not any given voter will cast a ballot, but knows that all participating voters will cast a ballot for the Republican party. In this case, the adversary can win the game we describe above by specifying a ballot value $\beta = \text{“Democrat.”}$

It is evident therefore that for any definition of coercion-resistance to be meaningful, the adversary must have uncertain knowledge about how—and indeed whether—some voters will cast their ballots. In other words, coercion-resistance requires that there be some “noise” or statistical uncertainty in the adversary’s view of voting patterns. To our benefit, it is natural to expect that in a real-world election an adversary can obtain only fragmentary knowledge about the likely behavior of voters. This means that coercion-resistance is a viable possibility.⁵ For a collection of n voters outside the control of the adversary—i.e., voters not subject to coercion—we characterize the view of the adversary in terms of a probability distribution D_{n,n_C} . We let ϕ be a symbol denoting a null ballot, i.e., an abstention, and let λ denote a ballot cast with an invalid credential. Then D_{n,n_C} is a distribution over vectors $(\beta_1, \beta_2, \dots, \beta_n) \in (n_C \cup \phi \cup \lambda)^n$, i.e., over the set of possible ballot choices for an election plus abstentions and invalid ballots. Thus, the distribution D_{n,n_C} serves the purpose in our experiment of defining the distribution of the “noise” that conceals the behavior of voters targeted by the adversary for coercion. For a set of n voting credentials $\{sk_i\}$, we let $\text{vote}(\{sk_i\}, PK_T, n_C, D_{n,n_C}, k_2)$ denote the casting of ballots according to distribution D_{n,n_C} . In other words, a vector $(\beta_1, \beta_2, \dots, \beta_n)$ is drawn from D_{n,n_C} and vote β_i is cast using credential sk_i .

We are now ready to present an experiment *c-resist* that defines the game described above between an adversary and a voter targeted for coercion. Recall that k_1, k_2 , and k_3 are security parameters defined above, n_V is the total number of eligible voters for the election, and n_C is the number of candidates, i.e., the size of the candidate slate. We let n_A denote the number of voters that may be completely controlled, i.e., corrupted by the adversary. We define $n_U = n_V - n_A - 1$. In other words, the number of uncertain votes n_U equals the total number of possible votes, minus those coming from voters controlled by the attacker, minus the vote coming from the voter the attacker is trying to coerce (in the experiment). Note that n_U is therefore the number of voters that contribute “noise” to the experiment.

We consider a static adversary, i.e., one that selects voters to corrupt prior to protocol execution. We assume that the adversary has a list of “voter names,” i.e., a roll of potential participating voters.

We let \leftarrow denote assignment and \leftarrow denote the append operation, while $\%$ denotes the beginning of an annotative comment on the experiment. Our experiment treats the case in which the adversary seeks to coerce a single voter; extension of the definition to coercion of multiple voters is straightforward. The experiments defined here halt when an output value is produced.

⁵ Additionally, it is possible for voting authorities—or indeed any entity—intentionally to inject “chaff” in the form of blank and invalid ballots into an election system.

Experiment $\text{Exp}_{\text{ES}, \mathcal{A}, H}^{c\text{-resist}}(k_1, k_2, k_3, n_V, n_A, n_C)$

```

 $V \leftarrow \mathcal{A}(\text{voter names, "control voters"});$  %  $\mathcal{A}$  corrupts voters
 $\{(sk_i, pk_i) \leftarrow \text{register}(SK_{\mathcal{R}}, i, k_2)\}_{i=1}^{n_V};$  % voters are registered
 $(j, \beta) \leftarrow \mathcal{A}(\{sk_i\}_{i \in V}, \text{"set target voter and vote"});$  %  $\mathcal{A}$  sets coercive target
if  $|V| \neq n_A$  or  $j \notin \{1, 2, \dots, n_V\} - V$  or
 $\beta \notin \{1, 2, \dots, n_C\} \cup \phi$  then % outputs of  $\mathcal{A}$  checked for validity
  output '0';
 $b \in_U \{0, 1\};$  % coin is flipped
if  $b = 0$  then % voter evades coercion
   $\tilde{sk} \leftarrow \text{fakekey}(PK_{\mathcal{T}}, sk_j, pk_j);$ 
   $\mathcal{BB} \leftarrow \text{vote}(sk_j, PK_{\mathcal{T}}, n_C, \beta, k_2);$ 
else % voter submits to coercion
   $\tilde{sk} \leftarrow sk_j;$ 
 $\mathcal{BB} \leftarrow \text{vote}(\{sk_i\}_{i \neq j, i \in V}, PK_{\mathcal{T}}, n_C, D_{n_U, n_C}, k_2);$  % ballots posted for honest voters
 $\mathcal{BB} \leftarrow \mathcal{A}(\tilde{sk}, \mathcal{BB}, \text{"cast ballots"});$  %  $\mathcal{A}$  posts to  $\mathcal{BB}$ 
 $(X, P) \leftarrow \text{tally}(SK_{\mathcal{T}}, \mathcal{BB}, n_C, \{pk_i\}_{i=1}^{n_V}, k_3);$  % election results are tallied
 $b' \leftarrow \mathcal{A}(X, P, \text{"guess } b\text{"});$  %  $\mathcal{A}$  guesses coin flip
if  $b' = b$  then % experimental output determined
  output '1';
else
  output '0';

```

The adversary \mathcal{A} in the above experiment is quite powerful, being capable (when $b = 1$) of complete coercion of the targeted voter. In order to characterize the success of \mathcal{A} , we must compare \mathcal{A} with a second adversary \mathcal{A}' . \mathcal{A}' is capable of coercion only within the framework of an ideal voting experiment *c-resist-ideal*. In other words, \mathcal{A}' characterizes the type of security against coercion that we would like to achieve in ES.

The main feature we are aiming for in our ideal experiment *c-resist-ideal* is for \mathcal{A}' to learn nothing from the private keys she acquires from corrupted players and from the coerced player. In particular, \mathcal{A}' cannot use private keys to perform active attacks. We cause \mathcal{A}' to express voting choices in a direct, ideal process; \mathcal{A}' cannot cast ballots, but merely enumerates the choices of players in her control. Additionally, \mathcal{A} cannot use private keys to learn information about the voting behavior of honest players or the coerced player. The *only* information that \mathcal{A}' gets is the grand total \mathcal{X} of votes in the election.

One feature of our experiment is counterintuitive. Because this is an ideal experiment, \mathcal{A}' is *always* given \tilde{sk} as the key of the coerced player. This is because \mathcal{A}' should be unable to determine, on the basis of keying material, from the situation in which coercion is successful or unsuccessful.

We require a function for the definition. We include here an ideal function *ideal-tally* that tallies the ballots posted to \mathcal{BB} in a special way. *ideal-tally* tallies in a normal manner all of the ballots cast by honest voters, i.e., prior to adversarial posting. The ballots cast by \mathcal{A}' , however, are treated specially. In particular, the function *ideal-tally* determines for each ballot B what the underlying private key sk_i is. If $i \notin V$, i.e., if the private key is not one assigned to one of the corrupted players, then the corresponding vote is not counted. Additionally, any double vote is not counted, i.e., *ideal-tally* performs the weeding of double votes that normally occurs during the tallying procedure.

Finally, ideal-tally does the following based on the value of the secret bit b . If $b = 0$, then ideal-tally does not count any ballot cast (by the adversary) using private key \tilde{sk} . If $b = 1$, then ideal-tally does include in the final tally a ballot cast using \tilde{sk} (excluding double votes).

Our definition of ideal-tally here assumes that every ballot has a unique corresponding private key. This is true of most natural ballot structures (and true of our proposed scheme). This definition, of course, also assumes ideal functionality in ideal-tally, namely the ability to extract private keys and plaintext votes from ballots. We do not specify in our definition how this “oracle” power is achieved. In our proofs, we construct a simulator capable of performing this functionality required from ideal-tally.

Note that although \mathcal{A}' learns the secret keys of voters, in our ideal experiment these secret keys in fact provide \mathcal{A}' with no information useful in voting—the ideal function ideal-tally ensures against misuse of keys—and no information useful in learning votes—because \mathcal{A}' never sees \mathcal{BB} .

We are now ready to present the experiment *c-resist-ideal* that characterizes the success of \mathcal{A}' .

Experiment $\text{Exp}_{\text{ES}, \mathcal{A}, H}^{c\text{-resist-ideal}}(k_1, k_2, k_3, n_V, n_A, n_C)$

$V \leftarrow \mathcal{A}'(\text{voter names, “control voters”});$	% \mathcal{A}' corrupts voters
$\{(sk_i, pk_i) \leftarrow \text{register}(SK_{\mathcal{R}}, i, k_2)\}_{i=1}^{n_V};$	% voters are registered
$(j, \beta) \leftarrow \mathcal{A}'(\text{“set target voter and vote”});$	% \mathcal{A}' sets coercive target
if $ V \neq n_A$ or $j \notin \{1, 2, \dots, n_V\} - V$ or $\beta \notin \{1, 2, \dots, n_C\} \cup \phi$ then	% outputs of \mathcal{A}' checked for validity
output ‘0’;	
$b \in_U \{0, 1\};$	% coin is flipped
if $b = 0$ then	% voter evades coercion
$\mathcal{BB} \leftarrow \text{vote}(sk_j, PK_{\mathcal{T}}, n_C, \beta, k_2);$	
$\tilde{sk} \leftarrow sk_j;$	
$\mathcal{BB} \leftarrow \text{vote}(\{sk_i\}_{i \neq j, i \in V}, PK_{\mathcal{T}}, n_C, D_{n_U, n_C}, k_2);$	% ballots posted for honest voters
$\mathcal{BB} \leftarrow \mathcal{A}'(\tilde{sk}, \{sk_i\}_{i \in V}, \text{“cast ballots”});$	% \mathcal{A}' specifies vote choices
$(\mathcal{X}, P) \leftarrow \text{ideal-tally}(SK_{\mathcal{T}}, \mathcal{BB}, n_C, \{pk_i\}_{i=1}^{n_V}, k_3);$	% election results are tallied
$b' \leftarrow \mathcal{A}(\mathcal{X}, \text{“guess } b\text{”});$	% \mathcal{A}' guesses coin flip
if $b' = b$ then	% experimental output determined
output ‘1’;	
else	
output ‘0’;	

4 A Coercion-Resistant Election Protocol

We are now ready to introduce our protocol proposal. We begin by describing the cryptographic building blocks we employ. Where appropriate, we model these as ideal primitives, as explained.

El Gamal: El Gamal [24] represents a natural choice of cryptosystem for our purposes, and is our focus in this paper. For reasons that we explain below, we will adopt a modified version of the basic El-Gamal scheme which can be seen as a simplified version

of the well known Cramer-Shoup [19] cryptosystem (only providing semantic security with respect to a passive adversary).

We let \mathcal{G} denote the algebraic group over which we employ El Gamal, and q denote the group order. For semantic security, we require that the Decision Diffie-Hellman assumption hold over \mathcal{G} [7][53]. A public/private key pair in El Gamal takes the form $(y(= g^x), x)$, where $x \in_U \mathbb{Z}_q$. We let \in_U here and elsewhere denote uniform, random selection from a set. The private key x may be distributed among the n_T players in \mathcal{T} using (t, n_T) -Shamir secret sharing [52] over $GF[q]$, for $t > n_T/2$. This private key may be generated by a trusted third party or via a computationally secure simulation of this process [25]. Each player then holds a public/private key pair $(y_i(= g^{x_i}), x_i)$, where x_i is a point on the polynomial used for the secret sharing. A ciphertext in El Gamal on message $m \in \mathcal{G}$ takes the form $(\alpha, \beta) = (my^r, g^r)$ for $r \in_U \mathbb{Z}_q$. For succinctness of notation, we sometimes let $E_y[m]$ denote a ciphertext on message m under public key y . To re-encrypt a ciphertext (α, β) , it suffices to multiply it pairwise by a ciphertext on $m = 1$, i.e., to compute a new ciphertext $(\alpha', \beta') = (y^r \alpha, g^r \beta)$ for $r' \in_U \mathbb{Z}_q$.

To decrypt a ciphertext (α, β) , the plaintext $m = \alpha/\beta^x$ is computed. To achieve a threshold decryption of ciphertext (α, β) , each active player i publishes a decryption share $\beta_i = \beta^{x_i}$. The value β^x , and thus m , may be computed using standard LaGrange interpolation. Player i may prove the correctness of its share using an NIZK proof of the form $PK\{s : \beta_i = \beta^s \wedge u_i = g^s\}$ —essentially two Schnorr identification proofs [49] with conjunction achieved using techniques described in, e.g., [17]. We omit many details in this description regarding the scheduling of these operations and the use of commitments to avoid adversarial bias. (The reader is referred to, e.g., [12][25] for some discussion of these issues in relation to key generation.)

We note that another possible choice of cryptosystem for our voting scheme is that of Paillier [43].

Selected Cryptosystem, Modified El Gamal: As mentioned before our modified version of the El Gamal cryptosystem can be seen as a simplified version of the Cramer-Shoup [19], method. It is rather straightforward to prove that the scheme is actually semantically secure under the decisional Diffie-Hellman assumption. The argument closely follows the one presented in [19]. Here we provided a sketched version of such an argument. Imagine there exists a probabilistic polynomial time algorithm A which can break the semantic security of the proposed scheme. Then our goal is to describe a different algorithm S (a simulator) which uses A to break the decisional DH problem. So assume S receives on input a quadruple (g_1, g_2, h_1, h_2) and has to determine if this is a DDH quadruple or not. S constructs the public key (for the M-El Gamal scheme) as follows. It chooses x_1 and x_2 at random and sets $h = g_1^{x_1} g_2^{x_2}$ the rest is unchanged.

What is different is the decryption procedure: On input $(A, B, C) = (g_1^r, g_2^r, h^r m)$, S retrieves the message m as $m = C \cdot (A^{x_1} B^{x_2})^{-1}$

Note that in this way the simulator can always decrypt (and the distribution of the key is perfectly indistinguishable from real).

Next when the adversary comes up with the two messages m_0, m_1 he wants to be challenged on S proceeds as follows. It flips a random (private) bit b , and encrypts m_b as follows

$$(h_1^{kx_1} h_2^{kx_2} m, h_1^k, h_2^k)$$

(where k is a random value)

Note that if the given quadruple is a DH one the ciphertext has the right distribution. This is because $h_1^k = g_1^{k'}$ and $h_2^k = g_2^{k'}$ for some k' and $(h_1^{x_1} h_2^{x_2})^k = h^{k'}$ (for the same k')

If, on the other hand, the given quadruple is not a DH one then it is easy to check that the A gains no information at all about the encrypted message (this is because this time to decrypt adv has to know the secret exponents x_1 and x_2 which remains information theoretically hidden by h).

Threshold cryptosystem with re-encryption: Our first building block is a threshold public-key cryptosystem CS that permits re-encryption of ciphertexts with knowledge only of public parameters and keys. The private key for CS is held by \mathcal{T} in our construction.

To describe our aim in the ideal, we would like any ciphertext E to be perfectly hiding. We would like decryption to be possible only by having a majority of players in \mathcal{T} agree on a ciphertext to be decrypted. We model this latter ideal property as in terms of a special decryption oracle denoted by $D\tilde{E}C$. We assume further that any decryption performed by $D\tilde{E}C$ is publicly verifiable.

Plaintext Equivalence Test (PET): A *plaintext equivalence test* (PET) [27,34] is cryptographic primitive that operates on ciphertexts in a threshold cryptosystem. The input to PET is a pair of ciphertexts; the output is a single bit indicating whether the corresponding plaintexts are equal or not. PET may be realized as an efficient distributed protocol that reveals no additional, non-negligible information about plaintexts. For a detailed description of efficient methods to perform this verification, along with proofs of the properties of the construction, see [34]. Rather than focusing on a specific embodiment of PET, we model the ideal properties of the primitive by means of an oracle denoted by $P\tilde{E}T$, and with the property of public verifiability.

Mix network: A (re-encryption) mix network (MN) is a distributed protocol that takes as input an ordered set $\mathbf{E} = \{E_1, E_2, \dots, E_d\}$ of ciphertexts generated in a cryptosystem like El Gamal that permits re-encryption. The output of MN is an ordered set $\mathbf{E}' = \{E'_{\pi(1)}, E'_{\pi(2)}, \dots, E'_{\pi(d)}\}$. Here, $E'_{\pi(i)}$ is a re-encryption of E_i , while π is a uniformly random, secret permutation. This is to say that MN randomly and secretly permutes and re-encrypts inputs. Thus, the special privacy property of a mix network is this: An adversary cannot determine which output ciphertext corresponds to which input ciphertext, i.e., which inputs and outputs have common plaintexts. Stated another way, an adversary cannot determine $\pi(j)$ for any j with probability non-negligibly better than a random guess. A number of mix network constructions have been proposed that offer privacy and robustness against a static, active adversary capable of corrupting any minority of the n players (servers) performing the mix network operation. Some of these constructions offer the additional property of *verifiability*. In other words, a proof

is output that is checkable by any party and demonstrates, relative to E and the public key of the ciphertexts that E is correctly constructed. It is convenient to conceptualize MN as an ideal primitive in terms of an oracle \tilde{MN} for MN with the property of public verifiability.

There are many good choices of mix networks for our scheme; some examples of such schemes are those of Furukawa and Sako [23] and Neff [38].

Proofs of knowledge: As sketched in the above descriptions, we make use of NIZK (non-interactive zero-knowledge) proofs of knowledge [6] in a number of places. We do not describe these tools in detail, as they are standard tools in the cryptographic literature. Instead, we refer the reader to, e.g. [17], for discussion of construction and logical composition of such protocols, and [11] for a notational overview and discussion of efficient realization. As is the usual case, our use of NIZK proofs enforces a reliance on the random oracle model in the security proofs for our scheme [4].

4.1 Our Proposed Protocol

Setup: The key pairs $(SK_{\mathcal{R}}, PK_{\mathcal{R}})$ and $(SK_{\mathcal{T}}, PK_{\mathcal{T}})$ are generated (in an appropriately trustworthy manner, as described above), and $PK_{\mathcal{T}}$ and $PK_{\mathcal{R}}$ are published along with all system parameters.

Registration: Upon sufficient proof of eligibility from V_i , the registrar \mathcal{R} generates and transmits to V_i a random string $\sigma_i \in_U \mathcal{G}$ that serves as the credential of the voter. Such credentials can be generated in a distributed threshold manner (as in [25]), with each active server of \mathcal{R} sending the voter V_i its credential. \mathcal{R} then adds $S_i = E_{PK_{\mathcal{T}}}[\sigma_i]$ to the voter roll L .⁶ The voter roll L is maintained on the bulletin board \mathcal{BB} and digitally signed as appropriate by \mathcal{R} .

We assume that the majority of players in \mathcal{R} are honest, and can thus ensure that the \mathcal{R} provides V_i with a correct credential. Nonetheless, it is possible for \mathcal{R} to furnish V_i with a proof that S_i is a ciphertext on σ_i . To enforce coercion-resistance in the case where erasure of secrets by voters is not automatic, a *designated verifier proof* [29] must be employed for this proof. We note that credentials may be used for multiple elections.

Candidate-slate publication: \mathcal{R} or some other appropriate authority publishes a candidate slate C containing the names and unique identifiers in \mathcal{G} for n_C candidates, with appropriate integrity protection. This authority also publishes a unique, random election identifier ϵ .

Voting: Voter V_i casts a ballot for candidate c_j comprising M-El Gamal ciphertexts $(E_1^{(i)}, E_2^{(i)})$ respectively on choice c_j and credential σ_i . In particular, for $a_1, a_2 \in_U Z_q$:

$$E_1^{(i)} = (\alpha_1, \alpha'_1, \beta_1) = (g_1^{a_1}, g_2^{a_1}, c_j h^{a_1}), E_2^{(i)} = (\alpha_2, \alpha'_2, \beta_2) = (g_1^{a_2}, g_2^{a_2}, \sigma_i h^{a_2}).$$

⁶ In our definitions above, we use the common terminology of private and public keys—with corresponding notation sk_i and pk_i —to describe the credentials associated with voters. Shifting from a general exposition to our specific protocol, we now use σ_i instead of sk_i to denote a voter credential, and S_i instead of pk_i to denote a public representation thereof. This change of notation aims to reflect the fact that voters do not employ a conventional form of public-key authentication in our scheme.

The first is a ciphertext on the candidate choice of the voter, the second a ciphertext on the credential of the voter.

Additionally, V_i includes NIZK proofs of knowledge of σ_i and c_j , a NIZK that α_i, α'_i have the same discrete logarithm with respect to basis g_1 and g_2 and also a NIZK proof that $c_j \in \mathcal{C}$, i.e., that c_j represents a valid candidate choice. The latter can be accomplished, for example, using a disjunctive proof that the ciphertext constitutes a valid encryption of a candidate choice in \mathcal{C} . These three NIZK proofs, which we denote collectively by Pf , may be accomplished efficiently using standard techniques. As is standard practice, the challenge values for Pf are constructed using a call to a cryptographic hash function, modeled in our security analysis by a random oracle $\tilde{O}W$. Input to $\tilde{O}W$ for these challenge values includes ϵ, E_1, E_2 and commitment values required for realization of the NIZK proofs. V_i posts $B_i = (E_1, E_2, Pf)$ to \mathcal{BB} via an anonymous channel.

Tallying: To tally the ballots posted to \mathcal{BB} , the authority \mathcal{T} performs the following steps:

1. **Checking proofs:** \mathcal{T} verifies the correctness of all proofs on \mathcal{BB} . Any ballots with invalid proofs are discarded. For the valid, remaining ballots, let A_1 denote the list of ciphertexts on candidate choices (i.e., the E_1 ciphertexts), and let B_1 denote the list of ciphertexts on credentials (i.e., the E_2 ciphertexts).
2. **Eliminating duplicates:** The tallying authority \mathcal{T} performs pairwise PETs on all ciphertexts in B_1 , and removes duplicates according to some pre-determined policy, using e.g., order of postings to \mathcal{BB} . When an element is removed from B_1 , the corresponding element (i.e., that with the same index) is removed from A_1 . We let B'_1 and A'_1 be the resulting “weeded” vectors. This is equivalent to retaining at most one ballot per given credential.
3. **Mixing:** \mathcal{T} applies MN to A'_1 and B'_1 (using the same, secret permutation for both). Let A_2 and B_2 be the resulting lists of ciphertexts.
4. **Checking credentials:** \mathcal{T} applies mix network MN to the encrypted list L of credentials from the voter roll. \mathcal{T} then compares each ciphertext of B_2 to the ciphertexts of L using PET. \mathcal{T} retains a vector A_3 of all ciphertexts of A_2 for which the corresponding elements of B_2 match an element of L according to PET. This step achieves the weeding of ballots based on invalid voter credentials.
5. **Tallying:** \mathcal{T} decrypts all ciphertexts in A_3 and tallies the final result.

How to cheat a coercer: One possible implementation of the function `fakekey` is simply for the coerced voter V_i to select and reveal a random group element $\tilde{\sigma}_i$, claiming that this is the credential σ_i . (If coerced multiple times – whether for one or more elections—the voter V_i would, of course, release the same value $\tilde{\sigma}_i$.) In addition, partial or full transcripts from the registration phase may be given to the adversary, depending on the scenario, as we now explain.

Upon receiving a claimed credential $\tilde{\sigma}_i$, the adversary would like to verify if it is correct. Let us consider the possibility of doing so under each of our three possible assumptions on the registration phase; in doing so, recall that we always assume that the adversary can corrupt only a minority of servers in \mathcal{T} , and so, will not be able to decrypt any of the semantically secure encryptions of credentials.

1. Assume that there is a mechanism forcing erasure of voter information no longer needed at the end of the registration phase, and that only a minority of servers in \mathcal{R} may be corrupted. At the end of the registration process, each voter will erase information specifying what part of the transcript leading to the credential σ_i he got from what registration server. Without proofs or transcripts from individual servers of \mathcal{R} , it is not possible for the adversary to verify the correctness of $\tilde{\sigma}_i$.
2. Assume that the adversary cannot corrupt *any* server in \mathcal{R} . As mentioned, the registration servers may if desired use designated verifier proofs to prove to each voter that the share they send is authentic (i.e., will be part of the recorded transcript S_i). While the voter will be convinced of these proofs, the adversary will not; in fact, he cannot distinguish between real such proofs and proofs simulated by V_i . Therefore, V_i can convincingly release full *simulated* transcripts from the registration phase, corresponding to a credential $\tilde{\sigma}_i$.
3. Assuming that the user knows what (minority of) servers in \mathcal{R} are corrupted, but is not necessarily able to erase data, he can present the adversary with registration transcripts that are consistent with the view of the servers he knows to be corrupted, but inconsistent (in terms of the real share of σ_i) with the view of the servers that are not. The latter transcripts will be accompanied by simulated designated verifier proofs. Since the adversary may only corrupt a minority of servers in \mathcal{R} , and a majority is required to compute the credential σ_i , there will be at least one share of σ_i that V_i can change to obtain a fake credential $\tilde{\sigma}_i \neq \sigma_i$, without the detection of the adversary.

5 Proving Coercion-Freeness

In this section, we provide a detailed outline for proof of the property of coercion-freeness in our proposed election protocol. (We do not consider correctness or verifiability here, as these are more standard properties, and the proofs are more straightforward.) For the purposes of this proof, we assume the use of the M-El Gamal cryptosystem over a preselected group \mathcal{G} of order q . The coercion-freeness of our scheme is dependent on the Decision-Diffie Hellman (DDH) assumption on \mathcal{G} . Briefly stated, this assumption states that no algorithm with running-time polynomial in the security parameters for \mathcal{G} can distinguish between the two distributions D and D' with non-negligible probability: Here, D is the distribution of tuples of the form (y_1, g_1, y_2, g_2) , where $g_1, g_2 \in_U \mathcal{G}$, $y_1 = g_1^x$, and $y_2 = g_2^x$ for $x \in_U \mathbb{Z}_q$; i.e., the pair (y_1, g_1) and (y_2, g_2) are related by a common exponent. D' is the distribution of random tuples, i.e., tuples of the form (y_1, g_1, y_2, g_2) , where $y_1, g_1, y_2, g_2 \in_U \mathcal{G}$. For detailed treatment of this assumption (expressed in an alternative, equivalent form), see, e.g., [7].

5.1 Assumptions

As explained above, we simplify our analysis by assuming ideal constructions for a number of components in our election protocol. Our aim in doing so is twofold: (1) Our protocol is flexible enough to accommodate a range of cryptographic building blocks from the literature and (2) We wish to retain a focus on the conceptual and definition

elements of our paper, and not on protocol details. Hence, we assume the availability of oracles for the four following cryptographic operations in our protocol: mixing, plaintext equivalence testing (PET), threshold ciphertext decryption, and calls to the one-way or hash function required for NIZK proofs. We denote these oracles respectively by $\tilde{M}N$, $\tilde{P}ET$, $\tilde{D}EC$ and $\tilde{O}W$. Although the functioning of these oracles should be clear from our protocol description, we present it again here:

- The oracle $\tilde{M}N$ performs exactly the same function as a mix network. It accepts as input an ordered list $\mathbf{E} = \{E_1, E_2, \dots, E_d\}$ of ciphertexts under the public key $PK_{\mathcal{T}}$ of the tallying authorities. Its output on \mathbf{E} is an ordered set $\mathbf{E}' = \{E'_{\pi(1)}, E'_{\pi(2)}, \dots, E'_{\pi(d)}\}$ for a secret, random permutation π , where $E'_{\pi(i)}$ represents a re-encryption of ciphertext E_i .
- The oracle $\tilde{P}ET$ takes as input a pair of ciphertexts (E, E') under $PK_{\mathcal{T}}$. It outputs a ‘1’ if E and E' have identical corresponding plaintexts, and outputs ‘0’ otherwise.
- The oracle $\tilde{D}EC$ takes as input a ciphertext E under $PK_{\mathcal{T}}$. It outputs the corresponding plaintext.
- The oracle $\tilde{O}W$ takes as input a query value in $\{0, 1\}^*$, and outputs a random value $\{0, 1\}^{k_4}$, where k_4 is a security parameter (that may depend on k_1, k_2 and k_3). The output of $\tilde{O}W$ is consistent, in the sense that a given input value always yields the same output value. This oracle may be viewed as the ideal embodiment of a cryptographic hash function.

Each of these oracles accepts publicly viewable input from all participating authorities (talliers). Each tallier may be thought of as having a publicly readable tape to which it may write input values for a given oracle; each tape contains a write portion for each time-step of the protocol, which we assume to be synchronous. At the end of a given timestep, an oracle produces output according to the following procedure. If a majority of talliers have furnished identical non-null values Z on their tapes, then the oracle processes input Z and yields the corresponding output. If there is no non-null majority input, then the oracle simply outputs the special symbol \perp . The requirement for majority input ensures that the protocol execution is determined by honest players, i.e., effectively reduces \mathcal{A} to an honest-but-curious adversary once the ballot-posting phase for the election is complete.

We additionally assume for simplicity that key setup and registration are performed by a trusted entity. Our proofs may be extended to accommodate more general assumptions in which these two processes are performed in a distributed manner.

5.2 Proof Overview

Recall that our definition of coercion-freeness revolves around a game played between an adversary \mathcal{A} and a voter targeted for coercion. The aim of \mathcal{A} is to guess which of the following two behaviors the voter has adopted during the execution of an election system ES: (1) The voter has divulged valid voting credentials and abstained from voting or (2) The voter has divulged fake credentials and cast a ballot. In order to demonstrate that ES possesses coercion-freeness, we must show that \mathcal{A} can guess successfully with probability only negligibly better than a weaker poly-time adversary \mathcal{A}' interacting with an ideal election system. This adversary \mathcal{A}' is passive, and its only input is the final tally

\mathcal{X} of votes cast by honest voters in the completed election plus T , the number of ballots eliminated for invalid associated credentials.

Our proof strategy is to construct a polynomial-time algorithm \mathcal{S} that takes a set of ballots W of honest voters and simulates the election system ES in the experiment *c-resist*. If the simulation is indistinguishable to \mathcal{A} from use of the true functional components of ES, and \mathcal{A} cannot cause the simulation to deviate from correct execution, then we see that \mathcal{A} learns nothing more than the correct election tally \mathcal{X} and the number of bad ballots T . This means in turn that \mathcal{A} is no more powerful than the ideal adversary \mathcal{A}' characterized in our experiment *c-resist-ideal*. Thus ES is coercion-free.

The inability of the adversary to cause deviation in the experiment from correct execution hinges on our oracle definitions, which require majority agreement on input values. Given this, we show that the simulation produced by \mathcal{S} is indistinguishable by \mathcal{A} from a real experimental execution of *c-resist* under the DDH assumption on \mathcal{G} . Our proof relies on the semantic security of M-El Gamal. In particular, we make use of the following, useful fact implied by the DDH assumption: A poly-time adversary that selects a plaintext m cannot distinguish between the distribution of M-El Gamal ciphertexts on m (A_1, A_2, B) and the distribution of triplets of the form $(\alpha_1, \alpha_2, \beta)$, where $\beta \in_U \mathcal{G}$ and α_1, α_2 are distributed exactly as (A_1, A_2) , with non-negligible probability (in the security parameters for \mathcal{G}). In consequence of this observation, it is possible for \mathcal{S} to simulate the election process by substituting *random ciphertexts*, i.e., random triplets of group elements, for the real ciphertexts that would be processed in a true execution of the experiment *c-resist*. In particular, \mathcal{S} can simulate the ballots of voters not controlled by \mathcal{A} with a list of random ciphertexts. Additionally, \mathcal{S} can simulate the oracle \tilde{MN} by setting its simulated output to a list of random ciphertexts. Under the DDH assumption, \mathcal{A} cannot distinguish between the random ciphertexts furnished by \mathcal{S} and the ciphertexts that would be processed in a true execution of ES.

5.3 The Simulation

We now outline the steps of the simulation of *c-resist* executed by \mathcal{S} . Throughout the simulation, according to the usual technique in the literature, \mathcal{S} maintains state for the simulated oracle \tilde{OW} so as to ensure consistency of output values. Let $W \in D_{n_U, n_C}$ represent a set of ballots input into the simulation as representing the posting of honest voters. At the very beginning the simulator receives a quadruple (g_1, g_2, h_1, h_2) which is either a Diffie-Hellman quadruple or a random one, according to some hidden bit d . More formally, $d = 1$ if the quadruple is a DH one and $d = 0$ otherwise. The goal of the simulator is to guess which situation is dealing with.

1. **Setup:** \mathcal{S} chooses uniformly and at random two elements $x_1, x_2 \in_U Z_q$ and sets $h = g_1^{x_1} g_2^{x_2} \bmod p$. \mathcal{S} publishes the public key (g_1, g_2, h) and also a randomized candidate slate $\mathcal{C} = \{c_i\}_{i=1}^{n_C}$ such that $c_i = g_1^{r_i}$ for $r_i \in_U Z_q$. (For technical reasons in our proof, we require that candidate identifiers here be random, rather than comprising the set $\{1, 2, \dots, n_C\}$.)
2. **Registration:** \mathcal{S} simulates the registrar \mathcal{R} , generating a set of credentials $\{\sigma_i = g_1^{s_i}\}$ for $s_i \in_U Z_q$. For the encrypted credential list L_0 , the simulator \mathcal{S} publishes a list of n_V ciphertexts (using a public key generated as above).

3. **Adversarial corruption:** The adversary \mathcal{A} selects a set V of n_A voters to corrupt, as well as a voter j for coercion and a target vote β . If any of these selections are invalid, i.e., if $V \neq n_A$ or $j \notin \mathcal{V} - V$ or $\beta \notin \mathcal{C} \cup \phi$, then the simulation is terminated.
4. **Coin flip:** A coin $b \in_U \{0, 1\}$ is flipped.
5. **Credential release:** \mathcal{S} gives \mathcal{A} the set of credentials $\{\sigma_i\}_{i \in V}$ as well as a credential σ for the targeted voter j . If $b = 1$, then \mathcal{S} gives $\sigma = \sigma_j$; otherwise σ is a random string.
6. **Honest voter simulation:** For each ballot element in W , the simulator posts a ballot consisting of two ciphertexts $(\alpha_{i,1}, \alpha'_{i,1}, \beta_{i,1}), (\alpha_{i,2}, \alpha'_{i,2}, \beta_{i,2})$. \mathcal{S} also furnishes the associated NIZK proofs of the form specified above. Since the associated challenges value comes from $\tilde{O}W$, and may therefore be predetermined by \mathcal{S} , the NIZK proof may be simulated using standard techniques. Let \mathcal{A}_0 be the list of these ballots. Let \mathcal{A}^* be the associated set of plaintext ballot choices in W for which the associated credential is correct, i.e., excluding λ elements.

The simulator creates the ciphertexts above as follows. For each ballot element in W , \mathcal{S} chooses two elements r_i, k_i at random in Z_q and sets $(\alpha_{i,1} = h_1^{r_i}, \alpha'_{i,1} = h_2^{r_i}, \beta_{i,1} = h_1^{r_i x_1} h_2^{r_i x_2} c_j), (\alpha_{i,2} = h_1^{k_i}, \alpha'_{i,2} = h_2^{k_i}, \beta_{i,2} = h_1^{k_i x_1} h_2^{k_i x_2} \sigma_i)$.

7. **Adversarial ballot posting:** The adversary \mathcal{A} posts a set of ballots \mathcal{B}_0 and associated NIZK proofs.
8. **Decryption of ballots posted by the adversary:** \mathcal{S} checks the NIZK proofs in \mathcal{B}_0 . Let \mathcal{B}_1 be the list of ballots with correct proofs. For each ballot in \mathcal{B}_1 and each credential in $\{\sigma_i\}_{i \in V} \cup \sigma_j$, the simulator decrypts using his own private key (see above).
9. **Tallying simulation:** \mathcal{S} simulates the behavior of honest tallying authorities. Since these are a majority, any deviating behavior by tallying authorities in the control of \mathcal{A} may be ignored. This part of the simulation proceeds as follows:

- (a) **Proof checking:** Let \mathcal{E}_0 denote the combined list of input ballots \mathcal{A}_0 and \mathcal{B}_0 . \mathcal{S} simulates the behavior of honest tallying authorities in rejecting all ballots with invalid associated NIZK proofs. Let \mathcal{E}_1 be the resulting ballot list.
- (b) **Eliminating duplicates:** Since no mixing has yet occurred, \mathcal{S} may simulate the elimination of duplicate ballots using its own decryption key. Let \mathcal{E}_2 be the resulting ballot list.
- (c) **Mixing:** \mathcal{S} simulates the oracle $\tilde{M}N$ as applied to \mathcal{E}_2 by outputting an equal-length list \mathcal{E}_3 of random ciphertext triples. Likewise, \mathcal{S} simulates the mixing of \mathcal{L}_0 by outputting an equal-lengthed list \mathcal{L}_1 of random ciphertexts.
- (d) **Checking credentials:** \mathcal{S} simulates the process of credential checking. In a true protocol execution, this would involve sequential comparison using $P\tilde{E}T$ between each ballot in \mathcal{E}_3 (more precisely, the credential ciphertext therein) and the ciphertexts in \mathcal{L}_1 . Either a match is found, in which case a ballot is deemed to be based on a valid credential, or else the list \mathcal{L}_1 is exhausted, and the ballot is rejected.

\mathcal{S} simulates the output of $P\tilde{E}T$ for this phase of the protocol using its own decryption key as before. Let \mathcal{E}_4 be the resulting ballot list.

- (e) **Decryption:** This is done straightforwardly.

Now if the adversary outputs a guess bit b' the simulator returns b' as his own guess for the decisional Diffie-Hellman challenge.

Observe that if the simulator's input is a Diffie-Hellman triplet (that is $d = 1$) then the simulation above is perfectly indistinguishable from the experiment $\mathbf{Exp}_{\mathcal{ES}, \mathcal{A}, H}^{c-resist}$.

As a matter of fact, assuming $g_1 = g, g_2 = g^a, h_1 = g^b, h_2 = g^{ab}$ for some g , any ciphertext of the form $(\alpha_{i,1} = h_1^{r_i}, \alpha'_{i,1} = h_2^{r_i}, \beta_{i,1} = h_1^{r_i x_1} h_2^{r_i x_2} m)$ is actually a valid one. Indeed $h_1^{r_i} = g^{br_i} = g_1^{br_i}, h_2^{r_i} = g^{abr_i} = g_2^{br_i}$ and $h_1^{r_i x_1} h_2^{r_i x_2} m = g^{br_i x_1} g^{abr_i x_2} m = g_1^{br_i x_1} g_2^{br_i x_2} m = h^{br_i} m$.

This means that

$$\Pr[\mathcal{S} = 1 | d = 1] = \Pr[\mathbf{Exp}_{\mathcal{ES}, \mathcal{A}, H}^{c-resist}(\mathcal{V}) = 1] = \mathbf{Succ}_{\mathcal{ES}, \mathcal{A}}^{c-resist}(\mathcal{V})$$

where we denoted with \mathcal{V} the view of the adversary.

On the other hand if the simulator's input is not a Diffie-Hellman triplet (that is $d = 0$) then the view produced by the simulation above does not give any information (in a strong information theoretic sense) about the votes posted by the honest parties. This is because, assuming $g_1 = g, g_2 = g^a, h_1 = g^b, h_2 = g^c$ for some $c \in_U Z_q$, one has that a ciphertext of the form $(\alpha_{i,1} = h_1^{r_i}, \alpha'_{i,1} = h_2^{r_i}, \beta_{i,1} = h_1^{r_i x_1} h_2^{r_i x_2} m)$ actually "masks" the message m perfectly. Indeed $h_1^{r_i} = g^{br_i} = g_1^{br_i}, h_2^{r_i} = g^{cr_i} = g_2^{c' r_i}$ and $h_1^{r_i x_1} h_2^{r_i x_2} m = g^{br_i x_1} g^{cr_i x_2} m = g_1^{br_i x_1} g_2^{c' r_i x_2} m = g_1^{br_i x_1} g_2^{br_i x_2} g_2^{c'' r_i x_2} m = h^{br_i} g_2^{c'' r_i x_2} m$.

This means that, in this case, the probability that the simulator outputs one is equal to the probability that the adversary outputs one in experiment $\mathbf{Exp}^{c-resist-ideal}$.

More formally

$$\Pr[\mathcal{S} = 1 | d = 0] = \Pr[\mathbf{Exp}_{\mathcal{ES}, \mathcal{A}, H}^{c-resist-ideal}(\mathcal{V}) = 1] = \mathbf{Succ}_{\mathcal{ES}, \mathcal{A}}^{c-resist-ideal}(\mathcal{V})$$

This means that

$$\mathbf{Adv}_{\mathcal{S}}^{\text{ddh}} = \Pr[\mathcal{S} = 1 | d = 1] - \Pr[\mathcal{S} = 1 | d = 0] = \mathbf{Adv}_{\mathcal{ES}, \mathcal{A}}^{c-resist}$$

under the Decisional Diffie-Hellman Assumption this quantity is negligible.

6 Conclusion

Beyond the fundamental properties of correctness and verifiability, an electronic election system can ultimately inspire confidence in voters only if it is well protected against criminal interference. We have sought here to define *coercion resistance* in the broadest possible manner, encompassing not just abuse of voting receipts, but randomization, forced-abstention, and simulation attacks. Our investigations also capture the fundamental statistical limits of adversarial interference in an election, showing how voters can achieve true protection only by concealment in larger population. Our proposed coercion-resistant voting scheme underscores that these limits may be within practical reach. The main limitation on the scalability of our scheme is its quadratic complexity. That said, since the initial publication [30] of the ideas presented here, Clarkson,

Chong, and Myers [16] have devised and implemented a refined protocol in a system called Civitas, and achieved good scalability by partitioning the population of voters. It is certainly conceivable that there exists a provably secure, coercion-resistant electronic voting scheme with lower complexity—perhaps even linear. Constructing one remains an open problem.

References

1. Proxyvote.com: Shareholder election website (2008), Referenced 2008 at <http://www.proxyvote.com>
2. Vote-auction (2008), Referenced 2008 at <http://www.vote-auction.net>
3. Baudron, O., Fouque, P.-A., Pointcheval, D., Stern, J., Poupard, G.: Practical multi-candidate election system. In: PODC 2001, pp. 274–283. ACM Press, New York (2001)
4. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: 1st ACM Conference on Computer and Communications Security, pp. 62–73. ACM, New York (1993)
5. Benaloh, J.C., Tuinstra, D.: Receipt-free secret-ballot elections (extended abstract). In: 26th ACM STOC, pp. 544–553 (1994)
6. Blum, M., De Santis, A., Micali, S., Persiano, G.: Noninteractive zero-knowledge. *SIAM J. Comput.* 20(6), 1084–1118 (1991)
7. Boneh, D.: The Decision Diffie-Hellman problem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 48–63. Springer, Heidelberg (1998)
8. Boneh, D., Golle, P.: Almost entirely correct mixing with applications to voting. In: Atluri, V. (ed.) ACM CCS '02, pp. 68–77. ACM Press, New York (2002)
9. Brands, S.: Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy. MIT Press, Cambridge (2000)
10. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)
11. Camenisch, J., Stadler, M.: Efficient group signature schemes for large groups. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 410–424. Springer, Heidelberg (1997)
12. Canetti, R., Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Adaptive security for threshold cryptosystems. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 98–115. Springer, Heidelberg (1999)
13. Canetti, R., Dwork, C., Naor, M., Ostrovsky, R.: Deniable encryption. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 90–104. Springer, Heidelberg (1997)
14. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM* 24(2), 84–88 (1981)
15. Chaum, D., Ryan, P.Y.A., Schneider, S.A.: A practical voter-verifiable election scheme. In: di Vimercati, S.d.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 118–139. Springer, Heidelberg (2005)
16. Clarkson, M.R., Chong, S., Myers, A.C.: Civitas: A secure voting system. Technical Report Technical Report 2007-2081, Cornell University Computing and Information Science (May 2007), Civitas Web site referenced 2008 at <http://www.cs.cornell.edu/projects/civitas>
17. Cramer, R., Damgard, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)

18. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 103–118. Springer, Heidelberg (1997)
19. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)
20. Chaum, D.: Secret-ballot receipts: True voter verifiable elections. *IEEE Security and Privacy Magazine* 2(1), 38–47 (2004)
21. Fujioka, A., Okamoto, T., Ohta, K.: A practical secret voting scheme for large scale elections. In: Zheng, Y., Seberry, J. (eds.) AUSCRYPT 1992. LNCS, vol. 718, pp. 244–251. Springer, Heidelberg (1993)
22. Furukawa, J.: Efficient, verifiable shuffle decryption and its requirement of unlinkability. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 319–332. Springer, Heidelberg (2004)
23. Furukawa, J., Sako, K.: An efficient scheme for proving a shuffle. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 368–387. Springer, Heidelberg (2001)
24. El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory* 31, 469–472 (1985)
25. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: The (in)security of distributed key generation in dlog-based cryptosystems. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 295–310. Springer, Heidelberg (1999)
26. Hirt, M., Sako, K.: Efficient receipt-free voting based on homomorphic encryption. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 539–556. Springer, Heidelberg (2000)
27. Jakobsson, M., Juels, A.: Mix and match: Secure function evaluation via ciphertexts. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 162–177. Springer, Heidelberg (2000)
28. Jakobsson, M., Juels, A., Rivest, R.: Making mix nets robust for electronic voting by randomized partial checking. In: Boneh, D. (ed.) USENIX 2002, pp. 339–353 (2002)
29. Jakobsson, M., Sako, K., Impagliazzo, R.: Designated verifier proofs and their applications. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 143–154. Springer, Heidelberg (1996)
30. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: Workshop on Privacy in the Electronic Society (WPES), pp. 61–70 (2005)
31. Karlof, C., Sastry, N., Wagner, D.: Cryptographic voting protocols: A systems perspective. In: USENIX Security, pp. 33–49 (2005)
32. Kiayias, A., Yung, M.: Self-tallying elections and perfect ballot secrecy. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 141–158. Springer, Heidelberg (2002)
33. Kohno, T., Stubblefield, A., Rubin, A.D., Wallach, D.S.: Analysis of an electronic voting system. In: IEEE Symposium on Security and Privacy, pp. 27–40 (2004)
34. MacKenzie, P.D., Shrimpton, T., Jakobsson, M.: Threshold password-authenticated key exchange. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 385–400. Springer, Heidelberg (2002)
35. Magkos, E., Burmester, M., Chrissikopoulos, V.: Receipt-freeness in large-scale elections without untappable channels. In: Schmid, B., et al. (eds.) First IFIP Conference on E-Commerce, E-Business, E-Government (I3E), pp. 683–694 (2001)
36. Michels, M., Horster, P.: Some remarks on a receipt-free and universally verifiable mix-type voting scheme. In: Kim, K.-c., Matsumoto, T. (eds.) ASIACRYPT 1996. LNCS, vol. 1163, Springer, Heidelberg (1996)
37. Moran, T., Naor, M.: Receipt-free universally-verifiable voting with everlasting privacy. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 373–392. Springer, Heidelberg (2006)

38. Neff, A.: A verifiable secret shuffle and its application to e-voting. In: Samarati, P. (ed.) ACM CCS 2001, pp. 116–125. ACM Press, New York (2001)
39. Neff, C.A.: Practical high certainty intent verification for encrypted votes (October 2004), Referenced 2008 at <http://www.votehere.com/vhti/documentation/vsv-2.0.3638.pdf>
40. Niemi, V., Renvall, A.: How to prevent buying of votes in computer elections. In: Safavi-Naini, R., Pieprzyk, J.P. (eds.) ASIACRYPT 1994. LNCS, vol. 917, pp. 164–170. Springer, Heidelberg (1995)
41. Okamoto, T.: An electronic voting scheme. In: Terashima, N., et al. (eds.) IFIP World Congress, pp. 21–30 (1996)
42. Okamoto, T.: Receipt-free electronic voting schemes for large scale elections. In: Christianson, B., Lomas, M. (eds.) Security Protocols 1997. LNCS, vol. 1361, pp. 25–35. Springer, Heidelberg (1998)
43. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
44. Parker, S.: Shaking voter apathy up with IT. The Guardian, December 11 (2001)
45. Paul, R.: Geneva brings quantum cryptography to internet voting. Ars Technica, October 12 (2007), Referenced 2008 at <http://www.arstechnia.com>
46. Associated Press. France's first online primary shows internet's unprecedented importance in presidential race. International Herald Tribune, January 1 (2007), Referenced 2008 at <http://www.ihf.com>
47. Ryan, P.Y.A., Schneider, S.A.: Prêt à voter with re-encryption mixes. In: Gollmann, D., Meier, J., Sabelfeld, A. (eds.) ESORICS 2006. LNCS, vol. 4189, pp. 313–326. Springer, Heidelberg (2006)
48. Sako, K., Kilian, J.: Receipt-free mix-type voting scheme. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 393–403. Springer, Heidelberg (1995)
49. Schnorr, C.-P.: Efficient signature generation by smart cards. Journal of Cryptology 4(3), 161–174 (1991)
50. Schoenmakers, B.: A simple publicly verifiable secret sharing scheme and its application to electronic voting. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 148–164. Springer, Heidelberg (1999)
51. Schoenmakers, B.: Personal communication (2000)
52. Shamir, A.: How to share a secret. Communications of the Association for Computing Machinery 22(11), 612–613 (1979)
53. Tsionis, Y., Yung, M.: On the security of elGamal based encryption. In: Imai, H., Zheng, Y. (eds.) PKC 1998. LNCS, vol. 1431, p. 117. Springer, Heidelberg (1998)

A Remark on Strong Verifiability

We set forth our definitions of correctness and verifiability in the body of the paper to meet the minimal requirements for a fair election and to achieve some measure of conceptual simplicity. These definitions are adequate for most election scenarios, but have a technical deficiency that may be of concern in some cases. In particular, our definitions allow for the possibility that a voter controlled by \mathcal{A} casts a ballot corresponding to vote β , but that the ballot gets counted as a vote for β' . Since \mathcal{A} can choose the vote cast by a controlled voter in any case, this technical deficiency only means that \mathcal{A} can potentially cause the votes of *controlled voters only* to change in the midst of the election process. It

does not provide \mathcal{A} with control of a larger number of votes. Most importantly, we note that this definitional weakness does not apply to our proposed protocol, which meets the stronger definition we now set forth.

Nonetheless, one can envisage some (somewhat artificial) scenarios in which stronger guarantees may be desirable. For example, \mathcal{A} might have the aim of causing the victor in an election to win by the slimmest possible margin. In this case, if \mathcal{A} controls a majority of \mathcal{T} , then \mathcal{A} might seek to decrypt all of the ballots cast in an election and alter the votes of controlled voters so as to favor the losing candidate.

We discuss now how our definition of verifiability may be modified to discount the possibility of this type of attack. (Analogous modifications may be made to the definition of correctness.) In particular, we can require that P be a proof that every tallied vote corresponds uniquely to a credential for which a valid ballot has been cast. For this, we require a natural technical restriction on vote. Let $\langle \text{vote}(\cdot) \rangle$ denote the set of possible outputs for the randomized function `vote` on a particular input. We require that an output ballot be wholly unambiguous with respect to both the vote β and the credential sk . That is, we require $\langle \text{vote}(sk_0, PK_{\mathcal{T}}, n_C, \beta_0, k_2) \rangle \cap \langle \text{vote}(sk_1, PK_{\mathcal{T}}, n_C, \beta_1, k_2) \rangle = \phi$ if $\beta_0 \neq \beta_1$ or $sk_0 \neq sk_1$.

To strengthen our definition of verifiability, we alter $\mathbf{Exp}_{\mathcal{ES}, \mathcal{A}}^{ver}(k_1, k_2, k_3, n_V)$ such that if the following conditions 1 and 2 are met, then the output of the experiment is '1'. Otherwise it is '0'.

1. $\text{verify}(PK_{\mathcal{T}}, \mathcal{BB}, n_C, \mathbf{X}, P) = '1'$
2. For every injective mapping $f : \langle \mathbf{X} \rangle \rightarrow Z_{n_V}$ one of two conditions holds:
 - (a) $\exists B : B \in \mathcal{BB}, B \in \langle \text{vote}(sk_i, PK_{\mathcal{T}}, n_C, \beta, k_2) \rangle, \forall j f(j) \neq i$
 - (b) $\exists \beta \in \mathbf{X} : f(\beta) = i, \forall B \in \mathcal{BB}, B \notin \langle \text{vote}(sk_i, PK_{\mathcal{T}}, n_C, \beta, k_2) \rangle$

Conditions 2(a) and 2(b) here respectively specify that the adversary has successfully defeated the verifiability of the system either by causing all of the valid ballots associated with a particular credential not to be counted or else enabling multiple votes to be tallied for a single credential.

Given use of a verifiable mix network, our proposed protocol meets this stronger security definition for verifiability.

Receipt-Free K -out-of- L Voting Based on ElGamal Encryption^{*}

Martin Hirt

ETH Zurich, Department of Computer Science
hirt@inf.ethz.ch

Abstract. We present a K -out-of- L voting scheme, i.e., a voting scheme that allows every voter to vote for (up to) K candidates from a set of L candidates. The scheme is receipt-free, which means that even a malicious voter cannot prove to anybody how he voted. Furthermore, the scheme can be based on any semantically secure homomorphic encryption scheme, in particular also on the modified ElGamal encryption scheme which does not allow for efficient decryption of arbitrary large messages (but is more efficient than Paillier's encryption scheme).

We note that in contrast to the standard setting with receipts, in a receipt-free setting a K -out-of- L voting scheme cannot be derived directly from a yes/no voting scheme.

Finally, we show that the voting protocol of Lee and Kim is not receipt-free, opposed to what is claimed in the paper.

1 Introduction

1.1 Problem Summary

The goal of an electronic voting protocol is to compute the sum of the votes of all entitled voters. In the simplest case, every voter can cast one of two possible votes (Yes/No-votes). More generally, every voter may vote for any K candidates out of a list of L candidates (K -out-of- L voting schemes). A secure voting protocol must (at least) satisfy the following fundamental properties:

- **ELIGIBILITY.** Only entitled voters are able to submit a vote (respectively, the votes of unauthorized voters are not counted), and they are able to submit only one single vote.
- **CORRECTNESS.** The tally that pops up at the end of the vote is the correct sum of all valid votes; invalid votes have no influence to the tally.
- **UNIVERSAL VERIFIABILITY.** Anyone can verify that the published tally is correct.
- **SECRECY.** It is infeasible to find out which voter has submitted which vote. Secrecy should also be satisfied for partial information on votes, as well as for relation between votes of several voters.

^{*} A preliminary version of this text can be found in [Hir01, Chapter 5].

- RECEIPT-FREENESS. The voter cannot obtain a receipt proving the vote he has cast.

The receipt-free property is required to prevent voters from selling their votes. Its importance is disputed within the voting community, as the problem of vote-selling can be seen marginal. However, in a classical voting scheme, the absence of vote-buying can never be demonstrated; even long after a vote, rumors about a vote-buying server cannot be resolved. This is in contrast to the correctness of the result, which can be proven at the end of the vote (using universal verifiability). Hence, to our mind, limited correctness (with universal verifiability) might be acceptable; limited receipt-freeness is not.

Receipt-freeness is not achievable without taking some additional assumption on the communication model (e.g., untappable channels, voting booth) and/or the trust model (e.g., trusted hardware tokens): Evidently, if the vote-buyer can read all communication channels, then the voter's initial randomness, secret-keys etc. are a verifiable receipt for the submitted vote (the vote-buyer can simulate the voter's behavior by using the correct voting program, and compare the communication with the effective communication seen on the channels). This receipt can even be made zero-knowledge for the vote-buyer by using standard techniques (the voter proves knowledge of a secret key matching his public key, some randomness, such that when applying the voting program, the effective communication is produced).

1.2 Contributions

We propose a construction for receipt-free voting protocols based on homomorphic encryption with the following advantages over previous voting protocols:

- GENERALICALNESS. The construction as well as the security proofs are generic in the underlying encryption scheme, and can equally be instantiated with Paillier's scheme [Pai99] or with the modified ElGamal scheme [EG84, CGS97]. Note that the latter is significantly more efficient with comparable security (Paillier requires a bigger field for the same level of security than ElGamal).
- GENERALITY. The new protocol supports K -out-of- L elections for arbitrary K and L . In contrast to most (even non-receipt-free) voting protocols in the literature, we do not have to adjust the security parameter of the underlying encryption scheme when L is large. Furthermore, this is the first receipt-free scheme supporting arbitrary large K without exponential complexity (the complexity of the new scheme is linear in L and independent of K).
- EFFICIENCY. The proposed voting scheme is more efficient than any receipt-free voting scheme in the literature. For K -out-of- L voting (for any K), it requires only three times more communication than the most efficient 1-out-of- L scheme which is *not receipt-free* [CGS97].

Note that the apparent idea for constructing K -out-of- L voting protocols, namely running L parallel instances of a 1-out-of-2 protocol and have each

voter prove that at most K instances contain a 1-vote [BY86], cannot generically be applied in a receipt-free model: For example, in the protocols of [SK95, HS00, BFP⁺01], the voter does not know the randomness used for encrypting his own vote, and hence he cannot prove any statement on the submitted vote(s).

The new protocol is constructed along the lines of the protocol of [CGS97]: A set of N authorities jointly set-up a secret-key/public-key pair, where the secret-key is shared among the authorities. Every voter then encrypts his vote under the public-key, the authorities compute the sum of all submitted votes with using the homomorphic property of the encryption function, and jointly decrypt (and prove) the tally by applying techniques from threshold cryptography. Receipt-freeness is achieved by techniques similar to those of [LK00, BFP⁺01]: Every voter must have his encrypted vote re-randomized by a *randomizer*. This randomizer can be a designated authority, or a piece of hardware given to the voter. The randomizer acts as an “observer” [CP92] establishing receipt-freeness, but cannot violate the secrecy or the correctness of the vote. More precisely, the randomizer does not learn the vote, hence cannot violate the privacy of the protocol. Furthermore, the randomizer must prove to the voter that the new encryption is really a re-randomization of the original encryption, hence he cannot violate the correctness of the protocol. However, a malicious randomizer could help a voter to sell his vote.

The security of the protocol is specified with respect to a fixed parameter t : The correctness of the computed tally is guaranteed as long as at least t authorities remain honest during the whole protocol execution, and the secrecy of each vote is guaranteed as long as no t authorities maliciously collaborate with each other. Vote-buying is disabled under the assumption that the randomizer does not collaborate with the vote-buyer, and that the vote-buyer cannot tap the communication between the voter and the randomizer. Therefore, we require that the vote-buyer cannot tap the channels between the voters and the randomizer. We stress that these additional assumptions are required solely for the receipt-freeness of the scheme; even when the randomizer cooperates with the adversary and/or the adversary can tap the channels between the randomizer and the voter, still our voting scheme provides all security properties of non-receipt-free voting schemes. Hence, receipt-freeness is provided as a strict add-on.

Finally, we analyze the security of the protocol of [LK00] and show that it is not receipt-free, in contrast to what is claimed in the paper.

1.3 Previous Work and Comparison

Secret-ballot voting protocols were first proposed by Chaum [Cha81], based on the idea of a mix-net. Cohen (Benaloh) and Fischer [CF85] and Benaloh [Ben87] suggested a voting protocol based on a homomorphic encryption function. The first voting schemes based on blind signatures and anonymous channels were proposed by Chaum [Cha89] and Fujioka, Okamoto, and Ohta [FOO92]. Later, many schemes based on these approaches were published [BY86, Ive91, PIK93, Sak94, SK94, CFSY96, CGS97].

The concept of receipt-freeness was first introduced by Benaloh and Tuinstra [BT94], where also a first receipt-free voting protocol based on homomorphic encryption is proposed. However, their main protocol turned out to be not receipt-free [HS00]. Another receipt-free voting protocol was proposed in [NR94], but as this scheme bases on generic cryptographic tools (like general zero-knowledge proofs) it is very inefficient. We mention that using incoercible multi-party computation [CG96] does not suffice to achieve receipt-freeness: Voters who *want* to sell their vote can use committed random bits in the set-up phase, and can then later prove their vote based on this randomness. In the sequel, we briefly compare our scheme with the most prominent receipt-free voting schemes in the literature.

Sako/Kilian [SK95]. This voting scheme is based on a mix-net channel. The scheme suffers under similar disadvantages as other mix-net voting protocols: it requires a high communication complexity in the mix (especially the cut-and-choose proofs), and the tallying process cannot be performed incrementally (the whole mixing load must be performed after the last vote has been cast). Furthermore, this scheme is vulnerable to the so-called randomization attack [Sch99]: The coercer can force a voter to vote randomly by instructing him which encrypted ballot to take from the generated list. In this scheme, receipt-freeness is assumed under the assumption of physically untappable channel. If an adversary could tap these channels, then not only he could violate the receipt-free property, but also the secrecy property. However, this drawback can be fixed.

Okamoto [Oka96, Oka97]. This scheme uses the blind-signature approach. It requires each voter to be active in three rounds, which is a significant disadvantage in practice. Receipt-freeness is achieved under the (rather demanding) assumption of untappable anonymous channels. An adversary who can violate this assumption can break both receipt-freeness and secrecy of the scheme. It seems unclear how to get rid of this drawback.

Hirt/Sako [HS00]. This protocol uses homomorphic encryption for tallying and a small mix-net for vote generation. This approach awards higher efficiency than the previous approaches. However, also this protocol is vulnerable to the randomization attack [Sch99]. Also this scheme relies on the assumption of untappable channels, and also in this scheme, tapping these channels violates the secrecy of the votes (can be fixed). Furthermore, this protocol implements only 1-out-of- L elections for small L (the computational complexity of decrypting the tally is exponential in L).

Lee/Kim [LK00], **Baudron et al** [BFP⁺01]. Recently, [LK00] introduced the idea of using a randomizer for achieving receipt-freeness. However, their protocol is insecure (cf. Appendix A). Independently, [BFP⁺01] proposed a receipt-free voting protocol based on randomizers, using Paillier encryption [Pai99] for secrecy and general diverted proofs [OO89] for receipt-freeness. Paillier encryption makes the scheme less efficient than schemes based on modified ElGamal (like ours): for achieving the same level of security, Paillier requires a bigger security parameter than ElGamal.

Furthermore, using general diverted proofs might also yield a high bit complexity; this is not analyzed in the paper. Finally, the protocol is limited to 1-out-of- L votes (in contrast to K -out-of- L votes), and for large L , the security parameter of the underlying encryption scheme must be increased, slowing down all computations.

2 Preliminaries

2.1 Σ -Proofs

A Σ -proof is a three-move special honest-verifier zero-knowledge proof of knowledge. This notion originates from the notion of a Σ -protocol, as introduced by Cramer [Cra96]. We call a Σ -proof *linear* if the verifier's test predicate is linear, i.e., the sum of two accepting conversations is accepting as well. Several Σ -proofs can easily be combined to a new Σ -proof proving knowledge of all (AND-combination) or either (OR-combination) of the witnesses. For the AND-combination, the protocols are run in parallel, but the verifier requests the same challenge for all parallel instances. For the OR-combination, again the verifier requests only one challenge, but the prover is allowed to split this challenge into one sub-challenge for each instance, where the sub-challenges must add up to the challenge. This allows the prover to run the simulator for all but one instance. Note that both the AND- and the OR-combination preserves linearity. Any Σ -proof can be made non-interactive by applying the Fiat-Shamir heuristics [FS86]. Details and formal definitions of Σ -proofs are omitted due to space restrictions.

2.2 Identification Scheme

For voter identification, we assume an identification scheme where the identification protocol can be written as a linear Σ -proof. One can easily verify that Schnorr's identification scheme [Sch91] satisfies this requirement. A voter's secret key is denoted by z_v , the corresponding public key by $Z_v = g^{z_v}$ for an appropriate generator g . Furthermore, in a model providing receipt-freeness, it is essential that each voter knows his own secret key, and this should be ensured by the underlying public-key infrastructure. A protocol for ensuring knowledge of the secret-key for Schnorr's identification scheme is given in [HS00].

2.3 Designated-Verifier Proofs

We will also make use of so-called designated-verifier proofs. A designated-verifier proof is a proof which is convincing for one particular (designated) verifier, but completely useless when transferred from this designated verifier to any other entity. The notion of designated-verifier proofs was introduced in [JSI96]. The key idea of designated-verifier proofs is to prove knowledge of either the witness in question, or of the secret key of the designated verifier. Formally, the proof will be constructed as the OR-combination of the proof in question and a proof of knowledge of the designated verifier's secret-key.

3 The Encryption Function

We first state the requirements on the encryption function, and then show that the two classical homomorphic encryption functions, namely modified ElGamal and Paillier, satisfy the requirements. For space limitations, the full descriptions have been deleted from this extended abstract.

3.1 Requirements

We consider a semantically-secure probabilistic public-key encryption function $E_Z : \mathbb{V} \times \mathbb{R} \rightarrow \mathbb{E}, (v, \alpha) \mapsto e$, where Z denotes the public key, \mathbb{V} denotes a set of votes, \mathbb{R} denotes the set of random strings, and \mathbb{E} denotes the set of encryptions. We write E instead of E_Z for shorthand. The decryption function is $D_z : \mathbb{E} \rightarrow \mathbb{V}, e \mapsto v$, where z denotes the secret key. Again, we write D instead of D_z . Note that the computational complexity of the decryption function D_z may be polynomial in the decrypted cleartext v . For arbitrary large v , decryption is not required to be feasible.

We assume that E is a group homomorphism, i.e., $E(v_1, \alpha_1) \oplus E(v_2, \alpha_2) = E(v_1 + v_2, \alpha_1 \boxplus \alpha_2)$ for the corresponding group operations $+$ in \mathbb{V} , \boxplus in \mathbb{R} , and \oplus in \mathbb{E} , respectively. Note that the group operation in \mathbb{V} must be modular addition, but the operations in the other groups can be arbitrary.

Furthermore, we require E to be q -invertible for a given $q \in \mathbb{Z}$ meaning that for every encryption e , the decryption v and the randomness α of qe can be efficiently computed, i.e., the function $D_q : e \mapsto (v_q, \alpha_q)$ such that $qe = E(v_q, \alpha_q)$ is efficient (given Z). Additionally, we require that there is a number $u \leq q$, large enough that $1/u$ is considered negligible, with the property that all integers smaller than u are co-prime with q , i.e., $\forall u' < u : \gcd(u', q) = 1$. This property will be used in the knowledge extractors of the Σ -proofs¹. Note that v_q must be 0 due to the semantic security of E and the group structure of \mathbb{V} . This notion of q -invertibility is inspired by the notion of q -one-way group-homomorphism of Cramer [Cra96, CD98].

Finally, we require the existence of verifiable distributed protocols for key generation and for decryption. Note that every encryption scheme can be turned into a threshold variant by applying techniques of general multi-party computations, but such an approach would be rather inefficient.

3.2 Modified ElGamal Encryption

The ElGamal encryption function [ElG84], modified according to [CGS97], enhanced with a threshold setup protocol and a threshold group decryption [Ped91], satisfies all above properties. When used over a finite field G with $|G| = q$ prime, then the encryption function is q -invertible, and we set $u = q$.

¹ More generally, it would be sufficient to assume that for a given large u , there exists an efficiently computable and invertible bijection from \mathbb{Z}_u onto a subset of \mathbb{Z}_q , where each element in this subset is co-prime with q .

What should still be mentioned here is that computational complexity for decryption is linear in the size of the cleartext. However, in the context of this work, this issue will not be a problem.

3.3 Paillier Encryption

Also the probabilistic encryption function of Paillier [Pai99], enhanced by threshold setup and decryption [EPS00, DJ01], satisfies all required properties. For an RSA modulus n , this encryption function is n -invertible, and let u be a large prime which is guaranteed to be smaller than the smaller prime factor of n (e.g., we let n be the product of two secret 512-bit integers, and let u be a fixed 511-bit prime).

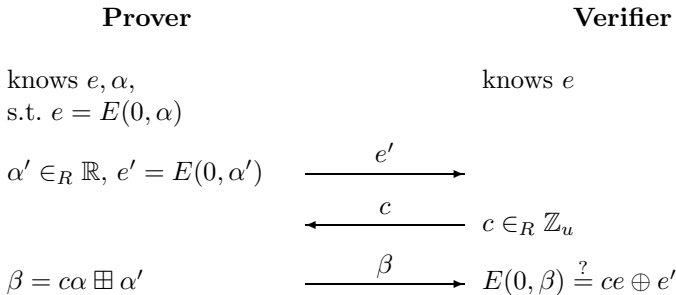
4 Re-encrypting and Proving Re-encryptions

A random re-encryption e' of a given encryption $e = E(v, \alpha)$ is an encryption with the same vote v , but a new (independently chosen) randomness α' . Such a re-encryption can be computed by adding a random encryption of 0 to e . Formally, a witness $\xi \in_R \mathbb{R}$ is chosen at random, and $e' = e \oplus E(0, \xi)$, i.e.,

$$e' = R(e, \xi) = e \oplus E(0, \xi).$$

Due to the homomorphic property of E , the randomness in e' is uniformly distributed over \mathbb{R} for a uniformly chosen $\xi \in_R \mathbb{R}$.

Proving that a given e' is indeed a re-encryption of e can easily be done by proving that $e' \ominus e$ is an encryption of 0. We present a simple linear Σ -proof for proving knowledge of a witness α such that $e = E(0, \alpha)$ for any given encryption e . The challenge for the protocol is uniformly selected from \mathbb{Z}_u , and the soundness of the protocol is proven under the assumption that E is q -invertible and that $\forall u' < u : \gcd(u', q) = 1$.



Completeness of the protocol is obvious by inspection. We next show that the protocol satisfies special soundness, by showing that if for any e' the prover can

reply to two different challenges $c_1 \neq c_2$, then he can compute a witness α with $e = E(0, \alpha)$. So assume that for two different challenges c_1 and c_2 , the prover can answer with β_1 and β_2 , respectively, such that both conversations (e', c_1, β_1) and (e', c_2, β_2) are accepting, i.e., $E(0, \beta_1) = c_1 e \oplus e'$ and $E(0, \beta_2) = c_2 e \oplus e'$, and hence $E(0, \beta_1 \boxminus \beta_2) = (c_1 - c_2)e$. Without loss of generality assume that $c_1 > c_2$, hence $0 < c_1 - c_2 < u$, and $\gcd(c_1 - c_2, q) = 1$. Hence we can apply the extended Euclidean algorithm to find two integers a and b such that $a(c_1 - c_2) + bq = 1$. Then, using the q -invertibility of the encryption function we compute α_q such that $q\alpha_q = E(0, \alpha)$. This results in

$$\begin{aligned} e &= (a(c_1 - c_2) + bq)e = a(c_1 - c_2)e \oplus bqe \\ &= aE(0, \beta_1 \boxminus \beta_2) \oplus bE(0, \alpha_q) = E(0, a(\beta_1 \boxminus \beta_2) \boxplus b\alpha_q). \end{aligned}$$

This concludes that indeed e encrypts 0 with witness $\alpha = a(\beta_1 \boxminus \beta_2) \boxplus b\alpha_q$.

We now show that the protocol is special honest-verifier zero-knowledge by constructing a simulator. The simulator is constructed as follows: For any given $c \in \mathbb{Z}_u$, we select β from \mathbb{R} at random, and set $e' = E(0, \beta) \ominus ce$. Obviously, the probability distribution of β is the same as the distribution of a real conversation in which α is chosen uniformly distributed (for the same challenge c).

It is important to note that the simulator can also be applied for an encryption e which does not encrypt 0, and the simulated conversation is computationally indistinguishable from a conversation where e encrypts 0 (an efficient distinguisher of these conversations would contradict the semantic security of the encryption function). This indistinguishability is important when several re-encryption proofs are OR-combined.

5 Non-Receipt-Free Voting Protocol

In this section we present a very simple K -out-of- L voting protocol which is *not* receipt free. The protocol is similar to the voting protocol of [CGS97], but due to a different ballot encoding it allows for votes with $K \geq 2$ and provides a substantially better computation complexity for $L > 2$. The protocol will be used as basis for the receipt-free protocol in the next section.

5.1 Model

We consider a model with N authorities A_1, \dots, A_N and M voters. Communication takes place by means of a bulletin board which is publicly readable, and which every participant can write to (into his own section), but nobody can delete from. The bulletin board can be considered as an authenticated public channel with memory. A threshold t denotes the number of authorities that is required for decrypting the tally, and which also is able to annihilate the secrecy of any vote.

5.2 Ballots

A ballot consists of a vector of votes, $\vec{v} = (v_1, \dots, v_L)$, where v_i is the vote for the i -th candidate. In a K -out-of- L election, a ballot is valid if and only if each vote v_i is either 0 or 1, and the votes on the ballot sum up to K . If voters should be allowed to vote for less than K candidates, then this is modeled as K -out-of- $(L + K)$ election, where the latter K candidates represent “abstain” and will not be tallied.

As simple notation, we write $E(\vec{v}, \vec{\alpha})$ for L -vectors $\vec{v} = (v_1, \dots, v_L)$ and $\vec{\alpha} = (\alpha_1, \dots, \alpha_L)$, meaning the component-wise application of the encryption function, i.e., $E(\vec{v}, \vec{\alpha}) = (E(v_1, \alpha_1), \dots, E(v_L, \alpha_L))$. Analogously, we defined $R(\vec{e}, \vec{\xi})$, $\vec{v}_1 + \vec{v}_2$, $\vec{\alpha}_1 \boxplus \vec{\alpha}_2$, and $\vec{e}_1 \oplus \vec{e}_2$.

5.3 Set-Up

In the set-up phase, the authorities jointly generate a uniformly distributed secret key and the corresponding public key for the encryption scheme, where the secret key is shared among the authorities, and the public key is publicly known. A protocol for (verifiable) generating a sharing of a randomly chosen secret key and a public key is a requirement on the encryption function.

5.4 Casting a Ballot

A ballot is cast as follows: The voter constructs a random encryption $\vec{e} = E(\vec{v}, \vec{\alpha})$ for his vote vector \vec{v} and randomness $\vec{\alpha} \in_R \mathbb{R}^L$, and posts it onto the bulletin board. Furthermore, the voter posts a proof of validity. A ballot $\vec{v} = (v_1, \dots, v_L)$ is valid if and only if $v_i \in \{0, 1\}$ for $i = 1, \dots, L$ and $\sum v_i = K$. In the following we construct a (finally non-interactive) validity proof for the encrypted ballot $\vec{e} = (e_1, \dots, e_L)$.

The validity proof is constructed as the AND-combination of a Σ -proof for each $i = 1, \dots, L$, each stating that e_i is an encryption of either 0 or 1, and a Σ -proof stating that $e_1 \oplus \dots \oplus e_L$ is an encryption of K . The proofs that e_i is an encryption of either 0 or 1 is constructed as an OR-combination of a proof stating that e_i encrypts 0 and a proof stating that e_i encrypts 1.

For easier notation, we write $e_{i,0} = e_i$ and $e_{i,1} = e_i \ominus E(1, 0)$, that is, e_{i,v_i} is an encryption of 0 with randomness α_i . Furthermore, we write $e_\Sigma = (e_1 \oplus \dots \oplus e_L)$, $\alpha_\Sigma = \alpha_1 \boxplus \dots \boxplus \alpha_L$, and $e_{\Sigma,K} = e_\Sigma \ominus E(K, 0)$. A ballot is valid exactly if for each i , either $e_{i,0}$ or $e_{i,1}$ encrypts 0, and $e_{\Sigma,K}$ encrypts 0. This proof can be constructed straight-forward as AND-combination of OR-combinations of proofs that a given encryption contain 0 (Section [4](#)).

The following protocol is a OR-combined Σ -proof of knowledge of a witness α_i such that $e_{i,0} = E(0, \alpha_i)$ OR $e_{i,1} = E(0, \alpha_i)$. In the protocol for proving $e_{i,1-v_i}$, the prover applies the simulator.

Prover**Verifier**

knows $v_i \in \{0, 1\}, \alpha_i$

knows $e_i = E(v_i, \alpha_i)$

$$\alpha'_{i,v_i} \in_R \mathbb{R},$$

$$e'_{i,v_i} = E(0, \alpha'_{i,v_i})$$

$$c_{i,1-v_i} \in_R \mathbb{Z}_u,$$

$$\beta_{i,1-v_i} \in_R \mathbb{R},$$

$$e'_{i,1-v_i} = E(0, \beta_{i,1-v_i})$$

$$\ominus c_{i,1-v_i} e_{i,1-v_i}$$

$$\xrightarrow{e'_{i,0}, e'_{i,1}}$$

$$\xleftarrow{c}$$

$$c \in_R \mathbb{Z}_u$$

$$c_{i,v_i} = c - c_{i,1-v_i} \pmod{u},$$

$$\beta_{i,v_i} = c_{i,v_i} \alpha_i \boxplus \alpha'_{i,v_i}$$

$$\xrightarrow{c_{i,0}, c_{i,1}, \beta_{i,0}, \beta_{i,1}}$$

$$c \stackrel{?}{=} c_{i,0} + c_{i,1} \pmod{u}$$

$$E(0, \beta_{i,0}) \stackrel{?}{=} c_{i,0} e_{i,0} \oplus e'_{i,0}$$

$$E(0, \beta_{i,1}) \stackrel{?}{=} c_{i,1} e_{i,1} \oplus e'_{i,1}$$

The finally validity proof is the AND-combination (i.e., parallel execution, but same challenge for all instances) of the above protocol for $i = 1, \dots, L$ plus a Σ -proof that $e_{\Sigma, K}$ encrypts 0. A (short) non-interactive proof is then the vector $[c, c_{1,0}, \dots, c_{L,0}, \beta_{1,0}, \dots, \beta_{L,0}, \beta_{1,1}, \dots, \beta_{L,1}, \beta_{\Sigma}]$ satisfying

$$\begin{aligned} c \stackrel{?}{=} & H \left(E(0, \beta_{1,0}) \ominus c_{1,0} e_{1,0} \parallel \dots \parallel E(0, \beta_{L,0}) \ominus c_{L,0} e_{L,0} \parallel \right. \\ & E(0, \beta_{1,1}) \ominus (c - c_{1,0}) e_{1,1} \parallel \dots \parallel E(0, \beta_{L,1}) \ominus (c - c_{L,0}) e_{L,1} \parallel \\ & \left. E(0, \beta_{\Sigma}) \ominus c e_{\Sigma, K} \right). \end{aligned}$$

The proof takes $3L + 2$ field elements.

5.5 Tallying

Tallying is performed for each candidate separately: For candidate i , the i -th components of each valid ballot are summed up (using the homomorphic property of the encryption function) and decrypted (using the verifiable decryption protocol of the encryption function). Note that it is known in advance that the decrypted tally will be in the range $(0, M)$; hence, decryption is efficient also for the modified ElGamal scheme.

5.6 Security Analysis

The privacy of the proposed protocol is guaranteed under the assumption that no t authorities maliciously pool their information, plus the assumption that the encryption function is semantically secure. The tally is correct if at least t authorities honestly participate in the tally decryption, plus the assumption that no verifier can cast an invalid ballot. The probability that an invalid ballot passes the validity proof is negligible if $1/u$ is negligible. The scheme is not receipt-free.

5.7 Efficiency Analysis and Comparison

We analyze the communication efficiency of this voting protocol for a K -out-of- L vote. The number of bits used to store one group element is denoted by B .

We ignore the costs for initialization and decryption of the final tally — they are independent of the number M of voters. It remains to count the costs for casting and proving votes. In order to cast his vote, every voter sends his encrypted ballot (LB bits) together with the validity proof ($(3L+2)B$ bits) onto the bulletin board. In total, $(4L+2)MB$ bits are posted to the bulletin board.

As comparison, in [CGS97], a ballot takes only B bits, but the proof takes $2LB$ bits. This gives a total of $(2L+1)MB$ bits. However, this scheme only allows for $K=1$ (for larger K the communication complexity would grow exponentially), and its decryption function is computationally inefficient for large L .

6 Receipt-Free Voting Protocol Based on Randomizers

In this section, the voting protocol of Section 5 is enhanced to be receipt-free. Therefore, the procedure for casting a vote must be modified.

The protocol relies on special authority called *randomizer*, who re-randomizes encrypted ballots of the voters. More precisely, each voter constructs an encrypted ballot containing his vote and secretly sends it to the randomizer. The randomizer re-encrypts this ballot and posts it to the bulletin board. Furthermore, the randomizer proves to the voter (in designated-verifier manner) that indeed the new encrypted ballot contains the same vote, and the voter and the randomizer jointly generate a proof of validity for this new ballot.

In the following, we briefly discuss the new model, then formally describe the new protocol for casting a ballot.

6.1 Model

In addition to the model of Section 5, we assume a special authority called *randomizer*. Collaboration of the randomizer with a vote-buyer or coercer cannot be tolerated. The randomizer does not learn the vote of any voter, nor can he interfere with the correctness of the tally, but he can reject to re-encrypt the ballot of any voter and thereby prevent this voter from participating the vote. Therefore, several randomizers can be used.

We assume that the communication channels between the voter and the randomizer are untappable for the vote-buyer. The privacy of these channel must be physical, in such a way that even the recipient cannot prove to the vote-buyer what was received from the channel (of course, the recipient can record all received data, but he must not be able to *prove* that he received a particular string). The untappable channels need not to be authenticated.

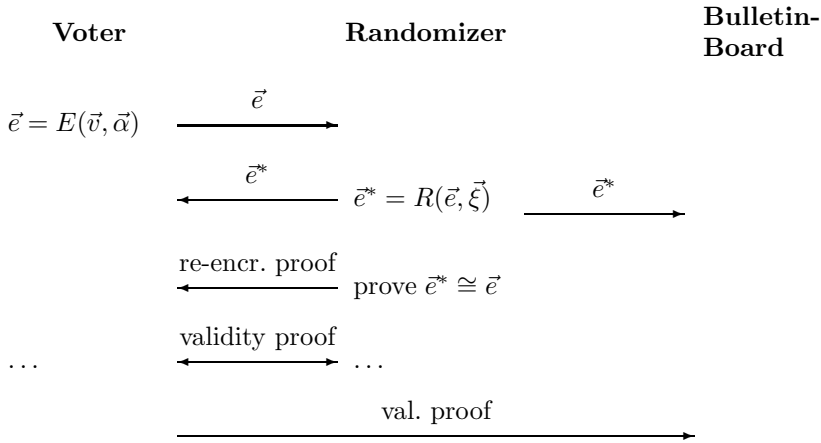
Furthermore, to each voter a secret key and a public key is associated, where the public key must be publicly known and the secret key must be kept private. We stress that in order to achieve receipt-freeness it must be guaranteed that

each voter knows the secret key corresponding to his known public key (but the voter is allowed to reveal the secret-key to the coercer). A protocol ensuring so is given in [HS00].

Note that all above requirements are uniquely relevant for receipt-freeness. If they are not met, then the proposed voting scheme still achieves all security requirements but receipt-freeness.

6.2 Casting a Ballot

A ballot is cast as follows: The voter constructs a random encryption $\vec{e} = E(\vec{v}, \vec{\alpha})$ of his vote vector \vec{v} with randomness $\vec{\alpha} \in_R \mathbb{R}^L$, and sends it through the untappable channel to the randomizer. The randomizer then computes random re-encryption $\vec{e}^* = R(\vec{e}, \vec{\xi})$ of \vec{e} , and proves to the voter in designated-verifier manner that indeed \vec{e}^* is a re-encryption of \vec{e} . Then, the voter and the randomizer jointly generate a validity proof for \vec{e}^* , without the randomizer learning anything about the vote vector \vec{v} , and without the voter learning anything about the re-encryption witness $\vec{\xi}$. Finally, the randomizer posts the validity proof to the bulletin board, and the voter posts the re-encrypted ballot \vec{e}^* .

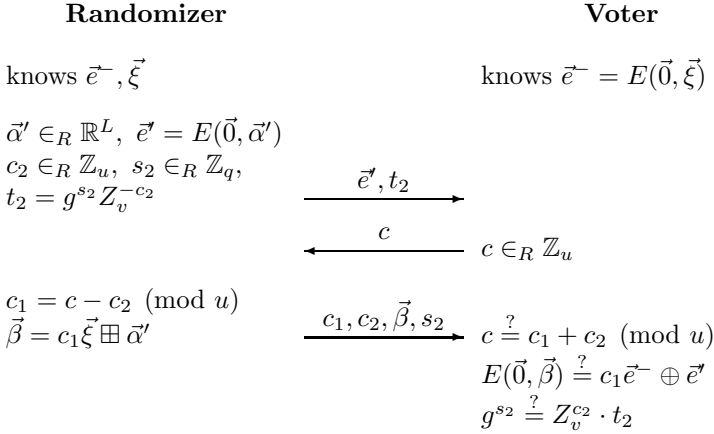


Designated-verifier re-encryption proof. The purpose of this proof is to have the randomizer prove to the voter that the new encryption \vec{e}^* is indeed a re-encryption of \vec{e} . However, this proof must be non-transferable, such that the verifier cannot convince someone else that \vec{e}^* is a re-encryption of \vec{e} . This is achieved by a designated-verifier proof (cf. Section 2.3): The randomizer proves knowledge of either a re-randomization witness $\vec{\xi}$ with $\vec{e}^* = R(\vec{e}, \vec{\xi})$, or of the voter's secret key. Obviously, this proof is convincing for the voter, but completely useless when transferred from the voter to a third party.

The proof is constructed as an OR-combination of the Σ -proof that the encryption $\vec{e}^* \ominus \vec{e}$ contains the vote $\vec{0}$ (which again is an AND-combination that $E(0, \xi_i) = e_i^* \ominus e_i$ for $i = 1, \dots, L$), and the Σ -proof of the identification scheme.

The resulting proof will require $L + 1$ encryptions in the first message, then one challenge, and in the final message, 2 sub-challenges plus $2L + 1$ randoms. Non-interactively, the proof can be made in $2L + 3$ field elements.

We show the proof for Schnorr’s identification scheme. We denote the voter’s secret key with z_v and the public key with $Z_v = g^{z_v}$. For shorthand, we set $\vec{e}^- = \vec{e}^* \ominus \vec{e}$. The following protocol is a Σ -proof of knowledge of either the voter’s secret key z_v satisfying $g^{z_v} = Z_v$, OR a witness $\vec{\xi}$ satisfying $\vec{e}^- = E(\vec{0}, \vec{\xi})$.



A non-interactive version of the proof is the vector $[c_1, c_2, \vec{\beta}, s_2]$ satisfying the equation

$$c_1 + c_2 \stackrel{?}{=} H\left(E(0, \beta_1) \ominus c_1 e_1^- \parallel \dots \parallel E(0, \beta_L) \ominus c_1 e_L^- \parallel g^{s_2} Z_v^{-c_2}\right).$$

This proof takes $L + 3$ field elements.

Validity proof. The validity proof is a non-interactive proof that the randomized encryption \vec{e}^* contains a valid vote, i.e., each e_i is an encryption of either 0 or 1, and in total, there are exactly K encryptions of 1. Neither the voter (who does not know the re-encryption witness $\vec{\xi}$) nor the randomizer (who does not know the ballot \vec{v}) can generate the proof on their own, hence they need to generate the proof interactively. The generation of the proof proceeds in two steps: First, the voter and the randomizer engage in an interactive protocol which gives to the randomizer a uniformly selected random non-interactive validity proof for \vec{e} (a so-called diverted proof [OO89]). Then, the randomizer adjusts this proof into a validity proof for \vec{e}^* .

Generating a diverted validity proof for \vec{e} . We first observe that validity proofs are linear Σ -protocols; hence, the sum of two accepting validity proofs (for the same vote \vec{e}) is again an accepting validity proof for \vec{e} . A diverted version of the validity proof can hence be generated as the sum of the normal validity proof (from Section 5.4) and a uniformly random validity proof for \vec{e} , generated by the

simulator. More precisely, a diverted proof for \vec{e} is generated as follows: First, the randomizer used the simulator to generate a random validity proof for \vec{e} with challenge 0 (here we use that the Σ -proof is special zero-knowledge). Then, the voter and the randomizer engage in an interactive validity proof for \vec{e} . The diverted proof is then the sum of these two proofs.

More precisely, the randomizer selects random “displacements” $c'_{i,0} \in_R \mathbb{Z}_u$, $c'_{i,1} = -c'_{i,0}$, and $\beta'_{i,0}, \beta'_{i,1} \in_R \mathbb{R}$ for $i = 1, \dots, L$. The displacements are chosen such that $c'_{i,1} + c'_{i,0} = 0$ for all i , i.e., the sum of the new sub-challenges will not change. Upon reception of the first message $(e'_{i,0}, e'_{i,1})$ of the interactive Σ -proof, the randomizer computes the first “message” of the non-interactive diverted proof as

$$e''_{i,0} = e'_{i,0} \oplus E(0, \beta'_{i,0}) \ominus c'_{i,0} e_{i,0}, \quad e''_{i,1} = e'_{i,1} \oplus E(0, \beta'_{i,1}) \ominus c'_{i,1} e_{i,1}$$

and asks as challenge $c = H(e''_{i,0}, e''_{i,1})$. When receiving the third message $(c_{i,0}, c_{i,1}, \beta_{i,0}, \beta_{i,1})$, the randomizer computes the third “message” $(c''_{i,0}, c''_{i,1}, \beta''_{i,0}, \beta''_{i,1})$ of the non-interactive diverted proof as

$$c''_{i,0} = c_{i,0} \boxplus c'_{i,0}, \quad c''_{i,1} = c_{i,1} \boxplus c'_{i,1}, \quad \beta''_{i,0} = \beta_{i,0} \boxplus \beta'_{i,0}, \quad \beta''_{i,1} = \beta_{i,1} \boxplus \beta'_{i,1}.$$

One can easily verify that the diverted conversation $((e''_{i,0}, e''_{i,1}), c, (c''_{i,0}, c''_{i,1}, \beta''_{i,0}, \beta''_{i,1}))$ is accepting for e_i (due to the linearity of the validity proof). Note that in the interactive validity proof, L such proofs are run in parallel with the same challenge (AND-combination). The above diversion is then applied on each parallel instance independently. Furthermore, as the original interactive proof is *honest-verifier* zero-knowledge only, one must ensure that the challenge of the randomizer is chosen at random. This is achieved by having the randomizer not only send c to the voter, but instead all $e''_{i,j}$, such that the voter can apply the hash function himself. Obviously, then the voter knows that the challenge is selected at random under the random oracle assumption.

Adjusting the diverted validity proof to \vec{e}^ .* With the above protocol, the randomizer can construct a diverted non-interactive validity proof for \vec{e} . It remains to convert this proof into a validity proof for \vec{e}^* . So consider the following diverted validity proof for \vec{e} : $[c, c''_{1,0}, \dots, c''_{L,0}, \beta''_{1,0}, \dots, \beta''_{L,0}, \beta''_{1,1}, \dots, \beta''_{L,1}, \beta''_{\Sigma}]$. Then one can easily verify that the following vector is a validity proof for the re-encrypted ballot $\vec{e}^* = \vec{e} \oplus E(0, \vec{\xi})$:

$$[c, c''_{1,0}, \dots, c''_{L,0}, \beta''_{\Sigma} \boxplus (\xi_1 \boxplus \dots \boxplus \xi_L), \beta''_{1,0} \boxplus c''_{1,0} \xi_1, \dots, \beta''_{L,0} \boxplus c''_{L,0} \xi_L, \beta''_{1,1} \boxplus c''_{1,1} \xi_1, \dots, \beta''_{L,1} \boxplus c''_{L,1} \xi_L].$$

6.3 Security Analysis (of the Vote-Casting Protocol)

The vote-casting protocol must satisfy two requirements: First, the randomizer must not learn the vote. Second, the voter must not be able to prove any correspondence between the original ballot \vec{e} and the re-encrypted ballot \vec{e}^* .

In order to show that the randomizer does not learn the voters vote, we only need to analyze the protocol for generating the diverted proof. This protocol is an interactive honest-verifier zero-knowledge proof of knowledge, which gives no information to the verifier (the randomizer) when the challenge is chosen honestly at random. Due to the modification that the voter applies the hash function by himself, it is clear that under the random oracle assumption the challenge is random, hence the protocol is zero-knowledge, and the randomizer learns nothing about the vote.

Secondly, in order to show that the protocol is receipt-free, we make use of two observations: First, the generated diverted validity proof is uniformly chosen among all validity proofs for \bar{e}^* , and second, the randomizer does not give any information beyond the diverted proof to the voter. The second observation can be verified by inspecting the protocol, but the first observation needs some more explanations: The diverted validity proof is the sum of the interactive proof as executed with the voter (and hence known to the voter), and a simulated proof which is selected completely uniformly among all accepting proofs (except for the challenge, which is random in the random oracle model). Hence the diverted proof is random and statistically unlinked to the interactive protocol that the voter is involved in. From the voter's viewpoint, the validity proof for \bar{e}^* is uniformly random and independent from all his own information.

Once more we stress that even a malicious randomizer cannot interfere with the secrecy or the correctness of the voting protocol. He only receives an *encrypted* ballot, and he must prove to the voter that the new ballot is a re-encryption of the original ballot.

6.4 Efficiency Analysis and Comparison

We consider K -out-of- L voting, and denote the number of bits per group element with B . As usual, we ignore the costs for initialization and decryption of the final tally.

In order to cast his vote, every voter sends the ballot to the randomizer LB bits, who sends a re-encryption and a re-encryption proof to the voter ($LB + (L + 3)B$ bits). Then, the voter and the randomizer run the interactive validity protocol ($(6L + 3)B$ bits), and the voter posts the randomized ballot (LB bits) and the randomizer posts the non-interactive proof to the bulletin board ($(3L + 2)B$ bits). This gives a total of $(9L + 6)MB$ bits sent through the untappable channels, and $(4L + 2)MB$ bits sent to the bulletin board.

In comparison, the 1-out-of- L voting protocol of [HS00] with N authorities and M voters requires $4LMNB$ bits sent through the untappable channels and $2L^2MNB$ bits posted to the bulletin board. For $K \geq 2$, this protocol has exponential communication complexity. Furthermore, the protocol has exponential computation complexity in L , and is hence applicable only for very small L .

Finally, we compare the proposed protocol with the 1-out-of- L voting protocol of [BFP⁺01]. The exact communication complexity of their protocol cannot be determined, as they do not provide a concrete diverted proof. As a rough estimate, the protocol communicates $18LMB$ bits over the untappable channels.

The size of their validity proof stored on the bulletin board is (according to their analysis) $(9L + 11)MB$. Furthermore, as they are restricted to Paillier encryption, they require a larger B than our scheme with ElGamal encryption for the same security level. Furthermore, they must require $B \geq L \log_2 M$ (a message must have enough bits for the tally of each of the L candidates), which for large L might require increasing B . Also their scheme cannot be used for $K \geq 2$; the size of the validity proof would grow exponentially.

6.5 Hardware Randomizer

In the proposed scheme, the randomizer essentially does not need to communicate with the bulletin board or the authorities (he can send the diverted validity proof signed to the voter, who then casts it on the bulletin board — a vote on the bulletin board is accept only if it is signed by the randomizer). This allows for a hardware-based receipt-free voting scheme: Every voter receives a personalized randomization-token, which performs the randomization of the vote, and generates a signed diverted validity proof for the randomized vote. Note that this randomization device acts as an “observer” [CP92]: It does not learn the vote, nor can it falsify it. Even when the vote authorities would distribute bad randomization tokens to the voter, still the privacy and the correctness of the vote would be guaranteed (but not the receipt-freeness). However, the device could reject to provide a proper validity proof; but in this case, the voter could demonstrate other people that his token is broken, and could get a new one.

7 Conclusions

We have presented a generic receipt-free voting scheme, which is secure with *any* homomorphic encryption scheme satisfying the required properties. There is no need to adapt the protocol and proofs to the encryption function, as is necessary for most voting schemes in the literature.

The resulting voting scheme is more efficient than any other receipt-free voting scheme. For K -out-of- L votes and N authorities, the communication complexity per voter is linear in L and independent of K and N . No other scheme in the literature has these properties.

For 1-out-of- L votes, the storage complexity on the bulletin board is the same that of the most efficient voting protocol which is not receipt-free [CGS97]. However, due to the communication with the randomizer, our communication complexity is about 3 times higher.

To the best of our knowledge, the presented scheme is the first scheme which can be based on ElGamal encryption without having a computation complexity growing exponentially in K . There are schemes with efficient computation also for large K , but they base on Paillier encryption [FPS00, DJ01]. Such schemes rely on stronger cryptographic assumptions and require larger security parameters, resulting in bigger constants in the computation and communication complexities.

References

- [Ben87] Cohen Benaloh, J.D.: Verifiable Secret-Ballot Elections. PhD thesis, Yale University (December 1987)
- [BFP⁺01] Baudron, O., Fouque, P.-A., Pointcheval, D., Poupard, G., Stern, J.: Practical multi-candidate election system. In: Proc. 20th ACM Symposium on Principles of Distributed Computing PODC (2001)
- [BT94] Cohen Benaloh, J.D., Tuinstra, D.: Receipt-free secret-ballot elections (extended abstract). In: Proc. 26th ACM Symposium on the Theory of Computing (STOC), pp. 544–553. ACM, New York (1994)
- [BY86] Cohen Benaloh, J.D., Yung, M.: Distributing the power of a government to enhance the privacy of voters. In: Proc. 5th ACM Symposium on Principles of Distributed Computing (PODC), August 1986, pp. 52–62 (1986)
- [CD98] Cramer, R., Damgård, I.B.: Zero-knowledge proofs for finite field arithmetic or: Can zero-knowledge be for free? In: Krawczyk, H. (ed.) CRYPTO '98. LNCS, vol. 1462, pp. 424–441. Springer, Heidelberg (1998)
- [CF85] Cohen Benaloh, J.D., Fischer, M.J.: A robust and verifiable cryptographically secure election scheme. In: Proc. 26th IEEE Symposium on the Foundations of Computer Science (FOCS), pp. 372–382. IEEE, Los Alamitos (1985)
- [CFSY96] Cramer, R., Franklin, M.K., Schoenmakers, B., Yung, M.: Multi-authority secret-ballot elections with linear work. In: Maurer, U.M. (ed.) EUROCRYPT '96. LNCS, vol. 1070, pp. 72–83. Springer, Heidelberg (1996)
- [CG96] Canetti, R., Gennaro, R.: Incoercible multiparty computation. In: Proc. 37th IEEE Symposium on the Foundations of Computer Science (FOCS), pp. 504–513 (1996)
- [CGS97] Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: Fumy, W. (ed.) EUROCRYPT '97. LNCS, vol. 1233, pp. 103–118. Springer, Heidelberg (1997)
- [Cha81] Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM* 24(2), 84–88 (1981)
- [Cha89] Chaum, D.: Elections with unconditionally-secret ballots and disruption equivalent to breaking RSA. In: Günther, C.G. (ed.) EUROCRYPT '88. LNCS, vol. 330, pp. 177–182. Springer, Heidelberg (1988)
- [CP92] Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO '92. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)
- [Cra96] Cramer, R.: Modular Design of Secure yet Practical Cryptographic Protocols. PhD thesis, CWI and Univ. of Amsterdam (November 1996)
- [DJ01] Damgård, I., Jurik, M.: A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In: International Workshop on Practice and Theory in Public Key Cryptography, PKC 2001 (2001)
- [ElG84] El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO '84. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)
- [FOO92] Fujioka, A., Okamoto, T., Ohta, K.: A practical secret voting scheme for large scale elections. In: Zheng, Y., Seberry, J. (eds.) AUSCRYPT '92. LNCS, vol. 718, pp. 244–251. Springer, Heidelberg (1993)

- [FPS00] Fouque, P.-A., Poupard, G., Stern, J.: Sharing decryption in the context of voting or lotteries. In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, pp. 90–104. Springer, Heidelberg (2001)
- [FS86] Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO '86. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
- [Hir01] Hirt, M.: Multi-Party Computation: Efficient Protocols, General Adversaries, and Voting. PhD thesis, ETH Zurich (2001); Reprint as ETH Series in Information Security and Cryptography, vol. 3, Hartung-Gorre Verlag, Konstanz (2001) ISBN 3-89649-747-2
- [HS00] Hirt, M., Sako, K.: Efficient receipt-free voting based on homomorphic encryption. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 539–556. Springer, Heidelberg (2000)
- [Ive91] Iversen, K.R.: A cryptographic scheme for computerized general elections. In: Feigenbaum, J. (ed.) CRYPTO '91. LNCS, vol. 576, pp. 405–419. Springer, Heidelberg (1992)
- [JSI96] Jakobsson, M., Sako, K., Impagliazzo, R.: Designated verifier proofs and their applications. In: Maurer, U.M. (ed.) EUROCRYPT '96. LNCS, vol. 1070, pp. 143–154. Springer, Heidelberg (1996)
- [LK00] Lee, B., Kim, K.: Receipt-free electronic voting through collaboration of voter and honest verifier. In: Japan-Korea Joint Workshop on Information Security and Cryptology (JW-ISC2000), pp. 101–108 (2000)
- [NR94] Niemi, V., Renvall, A.: How to prevent buying of votes in computer elections. In: Safavi-Naini, R., Pieprzyk, J.P. (eds.) ASIACRYPT '94. LNCS, vol. 917, pp. 164–170. Springer, Heidelberg (1995)
- [Oka96] Okamoto, T.: An electronic voting scheme. In: Proc. of IFIP 1996, Advanced IT Tools, pp. 21–30. Chapman & Hall, Boca Raton (1996)
- [Oka97] Okamoto, T.: Receipt-free electronic voting schemes for large scale elections. In: Christianson, B., Lomas, M. (eds.) Security Protocols 1997. LNCS, vol. 1361, pp. 25–35. Springer, Heidelberg (1998)
- [OO89] Okamoto, T., Ohta, K.: Divertible zero knowledge interactive proofs and commutative random self-reducibility. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT '89. LNCS, vol. 434, pp. 134–149. Springer, Heidelberg (1990)
- [Pai99] Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT '99. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
- [Ped91] Pedersen, T.P.: A threshold cryptosystem without a trusted party. In: Davies, D.W. (ed.) EUROCRYPT '91. LNCS, vol. 547, pp. 522–526. Springer, Heidelberg (1991)
- [PIK93] Park, C.-s., Itoh, K., Kurosawa, K.: Efficient anonymous channel and all/Nothing election scheme. In: Helleseth, T. (ed.) EUROCRYPT '93. LNCS, vol. 765, pp. 248–259. Springer, Heidelberg (1994)
- [Sak94] Sako, K.: Electronic voting schemes allowing open objection to the tally. Transactions of IEICE E77-A(1) (January 1994)
- [Sch91] Schnorr, C.P.: Efficient signature generation by smart cards. Journal of Cryptology (1991)
- [Sch99] Schoenmakers, B.: Personal communication (1999)
- [SK94] Sako, K., Kilian, J.: Secure voting using partially compatible homomorphisms. In: Desmedt, Y.G. (ed.) CRYPTO '94. LNCS, vol. 839, pp. 411–424. Springer, Heidelberg (1994)

- [SK95] Sako, K., Kilian, J.: Receipt-free mix-type voting scheme – A practical solution to the implementation of a voting booth. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT '95. LNCS, vol. 921, pp. 393–403. Springer, Heidelberg (1995)

A Analysis of the Lee-Kim Protocol

In this section, we show that the protocol of Kim and Lee [LK00] is not receipt-free, opposed to what is claimed in the paper.

A.1 Key Ideas of [LK00]

The protocol of [LK00] is based on the assumption of an *honest verifier* who ensures the validity of all cast votes. Each voter sends an encryption e of his vote to this honest verifier and proves its validity. Then, the honest verifier sends a random encryption e' of 0 to the voter and proves (with a three-move honest-verifier zero-knowledge protocol) that indeed e' is an encryption of 0. The final ballot of the voter is $e^* = e + e'$, which obviously contains the same vote as e , but different randomness. All communication between the voter and the randomizer must take place over an untappable channel.

Note that in this protocol a malicious “honest verifier” can help a voter to cast an invalid vote and thereby falsify the outcome of the whole vote. In our opinion, such a protocol in which the correctness of the tally relies on the trustworthiness of a single entity is questionable.

A.2 How to Construct a Receipt

The voter can easily construct a receipt: In the protocol where the honest verifier proves to the voter that indeed e' is an encryption of 0, the voter can choose the challenge as the output of a hash function applied to the message in the first move. This makes the transcript of the protocol a non-interactive proof (according to Fiat-Shamir heuristics) that e' is an encryption of 0. Hence, the values e' , e , the witness of e , and this proof are a receipt of the cast vote e^* .

A Secure Architecture for Voting Electronically (SAVE)

Jonathan A. Goler¹ and Edwin J. Selker²

¹ MIT

jagoler@berkeley.edu

² Excubate

ted.selker@gmail.com

1 Introduction

Electronic voting has the potential to be the most reliable, secure and trustworthy form of voting implemented. Digital technology, complete with error correction, robust storage and cryptographic security offers the possibility to record, transmit, store and tabulate votes far more reliably than paper. While current implementations of electronic voting have been susceptible to various failures, electronic voting itself is not fundamentally flawed. The Secure Architecture for Voting Electronically (SAVE) is one proposed architecture for mitigating security and trust issues with the voting process. In addition, the architecture enables academics, small companies and organizations to easily and cheaply build their own modules conforming to the standard.

Unfortunately, the first few examples of electronic voting machines have done little to inspire confidence in the technology. Early touchscreen systems (Direct Recording Electric or DRE) have suffered from poor user interfaces, system failures and data loss, resulting in voter frustration and distrust. One possible solution that is often presented as a solution to the trustworthiness of electronic voting systems is a Voter Verifiable Paper Trail (VVPT) [19] or more generally, Audit Trail (VVAT). VVATs are implemented as separate devices attached to, or observing the voting process, and indicating on a separate recording device, the selections of the voter.

Although the media has focused on recent failures of electronic voting systems, paper and mechanical systems have historically been easy to manipulate as well. Naturally, the term 'stuffing the ballot box' comes from the simple fraudulent addition of paper ballots. Computation actually enables better security through cryptographic means to ensure the propriety of votes cast and counted. In addition, electronic voting enables new classes of voting interfaces that would enable voters who have been discouraged from voting in the past.

Electronic voting systems present the opportunity to enfranchise many voters who would ordinarily have great difficulty voting [9,6,3,2]. Voting with assistive or speech interfaces as well as alternate means of ballot presentation could aid people with diminished motor capacity, visual impairments and even some cognitive impairments. If the general system of recording and processing votes is

separated from the user interface, then a greater variety of assistive user interface are possible to aid visually or physically impaired voters.

Changes in technology have solved or pushed security issues to different points in the election process. For example, lever machines simultaneously eliminated the individual ballots, which made counting less error-prone, while introducing the possibility of manipulation of the odometers on the machines. Both punch cards and optical-scan ballots permitted the ease of counting (and manipulation at the counting level) but maintained the both the logistical difficulty and integrity of individual ballots. So, large-scale fraud became possible, and if no alert was raised, the original count was not compared to the individual ballots.

DRE systems eliminated ballots and provided immediate feedback to the voter, allowing voters to correct problems with their votes. DRE feedback reduced the residual (voter error) rate to 0.4%, an improvement of over 50% over optical scan ballots [3]. However, the elimination of individual ballots prevents a meaningful audit trail.

A solution to security vulnerabilities posed by various mechanical and electrical voting systems is SAVE (Secure Architecture for Voting Electronically). SAVE is resilient to faults and malicious actors while maintaining voting secrecy requirements. The primary principle of SAVE is that there can be no single point of failure after the ballot leaves the control of the voter. It is modular, such that an election commission can select a variety of modules from vendors, assemble them, add the desired user interfaces and have a system that is inherently more reliable and resistant to both failure and attack. SAVE employs n-version programming, in which multiple versions of each module are written independently. The key advantage of this n-version architecture is that each independently written part is checking the others, ensuring that the system is well-behaved unless a very large percent of its modules are compromised.

Furthermore, in order to achieve the n-version program model and help ensure security, the system is broken into modules which each conduct a relatively simple operation.

2 Background

The voting process dates back to the Greeks, who used different pieces of pottery to indicate their votes for or against measures. The original voting *process* itself had several characteristics that made it desirable: the choices were obvious in that they were limited to a binary set, votes were represented by actual objects that could be seen and felt, and it was easy to verify the vote tally due to these two features. Today's voting scenario is far different from the direct democracy of the distant past. While typical ballots might have 12 races, some ballots contain in excess of fifty selections, and in the case of the 2003 California Recall, the candidate list for Governor was 135 candidates over six pages.

The current voting scene contains myriad technologies: lever machines from the 19th century, punch cards from the '60s, written ballots, optical scan ballots

and Direct Recording Electronic (DRE touch screens). While great controversy surrounded the punch card system in the 2000 election, there is a large and growing concern about the trustworthiness of DRE systems, precinct-counted optical scan and indeed all computers used in voting systems [15,19]. Clearly, given the history of continuous improvements in voting systems, new systems will emerge again in an attempt to ameliorate problems in previous technologies.

2.1 Security Requirements and Desires for the Voting Process

From an experience election researcher's point of view, new voting systems have a set of basic requirements that must be satisfied by any new system. For engineers, implementing these requirements poses new and unique challenges. Voting has a unique set of security requirements that are more complex and difficult to combine than other settings [27,20,28,21,22]. The basic security requirements of the voting process are:

1. Each voter must be verified to be permitted to vote on exactly the races for which they are permitted to vote, no more and no less.
2. Every vote cast must be counted as the voter intended.
3. The voter must not be able to prove that their vote was cast a particular way.

At the outset, the first two requirements are fairly straightforward. The first requirement involves the entire voting process, particularly the registration system and the polling station practices. The second requirement requires the user interface (paper, screen, touchpad) to respond properly to the desired selections, the chain of custody of that selection to be unbroken, and the final tally to accurately count each vote. The third requirement (to prevent vote buying or physical coercion) prevents a plaintext (Alice voted for Bob) receipt process.

2.2 Background Reliability of Electronic Devices

Computerization has been used with great success in the financial sector, a setting which demands absolute accuracy and reliability.

Computation systems are designed to be the most reliable systems for tabulation. By their very character, they are not subject to the kinds of mechanical failures that plague traditional voting equipment. Despite the advantages that electronic systems offer, several papers and well-known authors [15,19] have raised fears, uncertainties and doubts as to the effectiveness and trustworthiness of electronic voting equipment.

However, it is possible to create electronic voting systems that, by their very nature, are secure, reliable and trustworthy. An analysis of types of possible attacks, the possible scope of these attacks, and the likelihood that they will occur is a place to begin. The architecture should address these vulnerabilities.

3 Adversary Model

3.1 Unintentional Bugs and Physical Failures

On a systemwide basis, the largest likely contributor to failure in an electronic voting system is the unintentional failure of one of the components in the system. A monolithic system with one operating system, set of COTS (Common Off The Shelf) hardware, communication mode and voting software will suffer a catastrophic failure if a single component has a bug. While such failures may not be common, having a common failure mode may cascade and could render the entire system compromised. An example of such a large cascading failure is the 2003 northeast power failure [1], which started at a single failure point and affected the Eastern Seaboard, Midwest and Eastern Canada. In software, the blaster worm [4], caused serious outages throughout the world. Having diversity in the code of the voting system would help mitigate common failures and ensure that the vote can be properly counted even if some modules are compromised.

One concern about the Internet is that electronic transmissions can be held up or slowed down for one reason or another. A system that communicates electronically can batch the communication for later transmission, use land telephone lines to communicate the information, or use cell phones or satellite phones as alternate communications modes to make communication reliable. SAVE modules utilize both encryption and cached data so that disruption or compromise of the communication.

Additional hazards to the voting process include simple access to electronic power, and problems in transmitting votes from the polling stations. The dangers of power outages have successfully been addressed in Brazil where the computer-based voting system relies on batteries that last 14 hours. The question of messages being intercepted is one of simple encryption; the issue of changed messages would be dealt with using redundancy, cryptography and message authentication codes (MACs) to ensure integrity.

3.2 Intentional Manipulations

There are four groups of actors that we surmise would be interested in compromising the voting process.

The Evil Development Company. The danger of losing contracts due to faulty equipment has been a constant concern of election technology companies. They have small close-knit development organizations and review their work together. These are all safeguards for their systems. Still, there is concern that either as an individual or organization, the author of a voting system might insert malicious code. This code could change votes, delay or drop votes, or produce intentionally incorrect tallies. In addition, the code could flood the rest of the system with invalid messages, damaging the performance of the system. Finally, compromising elements such as specially designed cryptographic code might be inserted to leak information about

the election process to the otherwise secure communication channels. Included with this type of threat are the distributors of the code, as well as the hardware providers.

External Hackers. To date, external hackers have not had enough time and access to voting systems to hack them. Closed-Source voting systems such as Diebolds, which was found on an open FTP server in source code form, have appeared to have stark weaknesses [8,24,5,13]. That is, when Diebold's source code was exposed in this example, many vulnerabilities were easily visible to the programmers reviewing the code. With experience with the protocols and enough time— if a system is communicating over open lines— outside hackers could modify, delete and/or record messages between system components. If the system is not over an open network, this threat is of far less concern. Access to code would enable hackers to analyze the user interface and external ports for control codes that enable special modes in which votes can be changed, added or deleted. In the vast majority of voting systems which do not keep ballot images, the counts could easily be manipulated without recourse.

Malicious Voters. A voter gaining access to the system could try to vote more than once or as another person, or try to steal the votes of other individuals. Without gaining access to the system, voters may attempt to use phony smart-cards, claim/demonstrate that the phony card does not work and obtain a second valid card. While to date care has been taken to limit access to smart cards or other methods to opening a poll, it is possible and important to improve access control to the voting act.

Corrupt Election Officials. Election officials may be interested in more than running a fair election. Often such officials are political appointments, and as such may be subject to influence. In addition, poll workers may also have ulterior motives in their work. Thus, it is extremely important to design an architecture that would be resilient to and expose intentional fraudulent behavior on the part of election workers and officials.

By implementing multiple, diverse versions of each part of the voting system, as in SAVE, the evil development company suddenly can no longer compromise the entire voting process. External hackers and corrupt election officials have many more systems to analyze and compromise. Finally, malicious voters would now have to overcome a registration system that actually marks their ballot with an authentication code, preventing double voting.

3.3 Security of Paper Systems

Paper voting systems have a number of possible failure modes, as well as possible attacks. Even the best-practice methods of hand counting are more error-prone than electronic means, and most paper systems involve electronic scanning and tabulation [9]. They still present several attacks that must be anticipated, and countered. This section summarizes some attacks at various stages of the voting process.

Alteration of Ballots

Fraudulent Ballot. A paper ballot may be engineered to differ from the original ballot, with candidate names swapped, but without changing the coding on the ballot. This change would result in votes for a candidate A going to a different candidate B instead. This attack may be implemented on standard optical scanned ballots, or punch card ballots. This attack would be particularly successful if the target candidate has a large amount of support in a particular precinct, the ballot can be tampered with and the votes diverted. Countermeasures included validation of counts using small sample sets and verifying manually that each type of ballot mark is counted the same as the ballot text and markings should indicate.

Scanner Control Cards. Scanner control cards can be manipulated to achieve the same effect as fraudulent ballots. With control cards, the punches representing a vote for a candidate can be reprogrammed to vote for a different candidate. Counting testing procedures can be implemented to validate, using properly marked ballots (assuming they are not compromised), that control cards are properly marked.

Pre/Post-Voting Ballot Invalidation. After ballots are completed, a nefarious election worker may spoil ballots by adding overvotes or extraneous marks to the ballot. The opportunity for this attack is present both at central counting locations and at precinct counting locations. A ballot worker out of sight may mark ballots using a writing utensil or in sight could use even a clump of graphite hidden on the underside of a ring or fingernail to selectively invalidate ballots [16]. A two-man system where two workers are required to be present for the counting/moving or deposit of ballots would help alleviate this attack.

Destruction/Replacement of Ballots

Denial of Service. Pre-election, ballots can be spoiled via a variety of means including water damage, spilling ink, and surreptitious marking of ballots. These actions may result in spoiled ballots, and denial of voting rights to voters. While polling place operations could move voters to another precinct, the action may cause significant voter falloff for voters who are unable or unable to make it to an alternate site [23].

Post-Election. A more direct and effective means of tampering with the ballots would be simply to lose, 'misplace' or selectively damage ballots. An election worker may selectively invalidate (and replace) ballots to keep them from being counted properly. Effective countermeasures include placing digital signatures and/or serial numbers on ballots and recording those signatures and serial numbers along with a tally of the ballots passed out. Additional collusion on the part of polling place workers would help invalidate or even take advantage those countermeasures. Altering the serial numbers on the register may cause valid ballots to be invalidated, or altering the count would make the election appear invalid. Keeping careful records of ballot serial

numbers may help ameliorate this attack, and combined with a secure hashing algorithm, may provide public verifiability.

4 The SAVE Architecture

Our aim is to outline the architecture and principles of the SAVE system, rather than describe in detail its implementation. The architectural overview will illustrate the improvements and advantages over existing paper and electronic systems.

The architecture is composed of five principal layers: A *User Interface* and the *Listeners* which ensure proper capture of votes, the *Registration* to assure that the user is valid, the *Witnesses* layer to create an auditable and secure record, and *Aggregators* to establish an actual outcome. Additionally, feedback layers give the voter proof that the vote was established and recorded, as well as another layer between the registration systems and the aggregators, known as a mix-network, which can perform random secure shuffles of ballots to further guarantee anonymity in the final count.

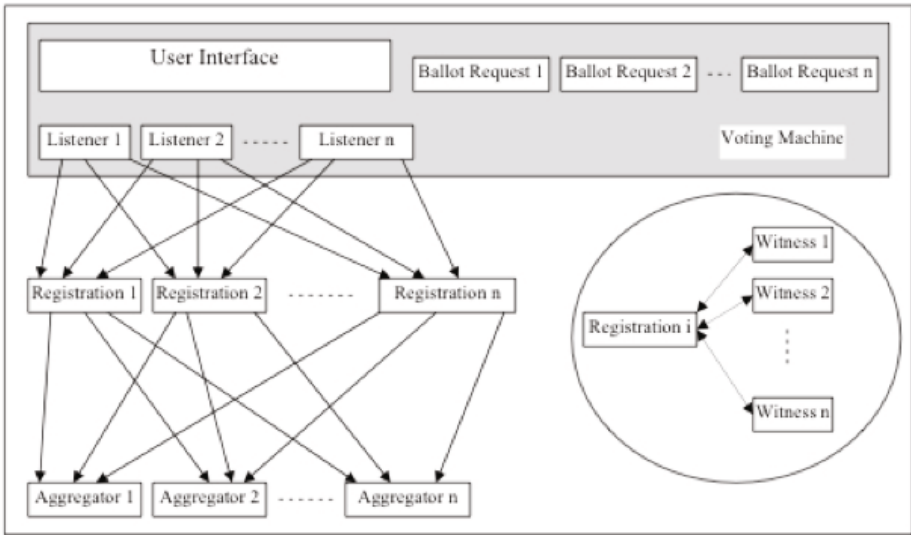


Fig. 1. The Basic Architecture of the SAVE system: A user interface (UI) is the only single point of failure in the system. Beyond the UI, multiple modules process each ballot.

For communication, SAVE uses the eXtensible Markup Language (XML) and an associated communication layer known as Simple Object Access Protocol (SOAP) [7]. XML and SOAP are a set of protocols that are available on all modern computer platforms. They are human readable, which means the commands and much of the data is text that can be read and understood, and they

allow for definitions of modules by virtually all programmers. Communication between the components is provided by an SOAP, an XML based messaging protocol.

Each level of the architecture logs the incoming and outgoing messages to aid in auditing the system. The modules are split up into small parts so that each of them contains fewer than 1000 lines of program code. This granularity will enable a faster and more thorough review process, limiting the number of bugs that can be introduced, while allowing enough space for code diversity.

4.1 Cryptographic Protocols

The entire process includes cryptographic protocols at appropriate stages: the Listeners read the input from the User Interface, and then encrypt and sign ballots; the registration module authenticates the voter and signs the ballot; the witnesses sign the ballot; the aggregators perform secure shuffles on the ballots and tally the results. Chaum [12,11] showed that it is possible to prove that a vote appears in the tally without proving what the vote is, thus the aggregation of ballots are published and a voter can verify that their vote is in the tally.

In addition to these fairly standard ways of protecting voting data, the system employs a more specific set of schemes for protecting it. When sending a ballot to a registration system, the architecture must assure that the voter is valid. The registration system, on the other hand, should have no knowledge of how the vote was cast. The filled ballot needs to be separated from the access to vote. Encryption additionally prevents others from seeing the voter's vote through the registration system. For governments that keep ballot data together a voter might vote absentee and also try to vote at a polling place. This type of fraud can be eliminated by the following procedure: take a ballot, encrypt the vote, send the vote along with the registration data to the registration server, and have the registration server return the same encrypted ballot, but with a signature attached. That signature is known as a blind signature. What is known as the "blinding factor" is an additional layer of encryption that the voter can decrypt [26,10,14]. It permits the system to obtain a signature for a plain text ballot even if the signer does not know the plain text. It is analogous to putting a piece of carbon paper in a sealed envelope and having someone sign the outside; the signature will appear on the inner piece of paper. Blind signatures maintain the privacy of the vote (not associated with the voter) while still proving that the vote was cast through the correct process.

In addition to the registration server signing and validating the ballot, a number of other modules must sign the ballot as redundant verification. These are known as witness modules, and various watchdog organizations as well as the political parties could provide them. They could be smart cards or pieces of software, and they all would provide a blind signature to ballots that are considered valid. In that way, when ballots are recorded, they are recorded with the signatures of all of these witnesses. The witness scheme permits an additional set of independent modules the ability to mark ballots as bona-fide, and thus provides enhanced verifiability and trustworthiness.

Aggregators simply decrypt, verify, store, and count ballot information. Care must be taken to make sure that they properly validate registration and witness signatures on the ballots and that they are properly placing the data into the repositories that they should be in. Having multiple aggregators allows us to “recount” on the fly. Aggregators provide redundancy of data and verify the entire process up to that point.

4.2 N-Versioning

The principle of redundancy with each module verifying the others is central. It enables the system to continue to work even if there is a failure somewhere along the line. Having multiple programs that process each stage of the ballot casting can establish improved reliability. Consider two modules running on two servers. Two modules compiled by different compilers, linked by different linkers, loaded by different loaders, being run on different operating systems, communicating with different communication stacks would have different errors in them, and be vulnerable only to different attacks. Because these versions can be transmitting over different networks, the system is more reliable. Because these are different programs, subverting one of them would not affect the others and still would ultimately enable an accurate vote to be cast. More importantly, if different people and organizations write these modules, intentional tampering of one module (discussed as the Evil Equipment Developer), such as putting in an Easter egg (a secret module of code that invokes undocumented functionality), would not affect the integrity of other modules.

N-version coding is currently used on a variety of critical applications. In particular, vital systems on aircraft, and military equipment use n-version systems in all layers of implementation, including sensors, software and hardware. In the case of nuclear weapons, even the humans involved conform to n-version principles (the two-man rule).

N-versioning on the actual code can only take security so far. While it can avoid common flaws amongst the modules, common flaws in the underlying hardware and operating system might still compromise the code for all modules. Thus it is essential to place different modules on different pieces of hardware, preferably with small, real-time operating systems, not large complicated systems such as Windows.

Certainly, to be sure these new measures are effective, the system will have to be tested beforehand. By forcing each module to comply with the specified abstraction-function behavior, the architecture will be uniformly black-box testable to ensure compliance with the protocol. Clearly, any certification of the system must include a thorough, formal review of the code. In addition, there must be no difference between a test vote and a real vote as far as the software is concerned.

The SAVE system separates the aspect of user interface from the rest of the voting system. The intent is to allow user-interface designers to build the best possible user interface for every type of user. A user who is blind might use a different interface than a user who has little motor control. This separation

of the user interface is an important one because user interfaces can be very complicated to build and maintain. The amount of code needed to write them is far larger than that required to perform cryptography or aggregation. However the user interface is implemented, that piece of software must communicate with the rest of the SAVE architecture in the same manner.

Once the user has filled out the ballot, the next step is to authenticate the voter. A back-end system checks the person's name against a database of registered voters. The registration server signs the vote, along with multiple electronic witnesses as described above. The witnesses sign the vote to indicate that a valid voter, as assessed by the registration server, cast it. At this point the signed, blinded ballots are then sent to a variety of aggregation servers to be counted.

Clearly one of the most serious concerns raised about n-versioning is the ability to truly diversify code. For simple operations there are a limited number of options a developer might choose. Developers may take similar overall approaches; they may use different languages, break their code into different functional blocks and write code in a particular style. Certainly a programmer introducing secret functionality would be diverse from others. Differences in the way code is written, just as for genetic diversity, leads some modules vulnerable to attacks that others are safe from and vice-versa.

5 Security Analysis of SAVE

The SAVE architecture assumes that all modules are using the best available standard encryption algorithms. The main security and reliability advantages of SAVE come from its redundancy and overall modular structure. One of the primary assumptions of SAVE is the independence of the code itself and the reliability of the platform in general.

Consider n modules at each of m stages, $M_{n,m}$, each with an internal failure rate $F_{n,m}$, and attack susceptibility of $A_{n,m}$. In addition, we must include inter-module communication channels between two modules a and b in different stages, $C_{a_m,b_{m+1}}$, for all a and b in different stages. To begin, we exclude the communication channel here, but include it below.

$$F_M = 1 - \prod_{m_1, m_2, \dots, m_n} (1 - F_{n,m})(1 - A_{n,m}) \quad (1)$$

Thus, for the SAVE system, the total possible rate of failure in any module (considering independence follows directly as shown in Equation 1).

However, SAVE implements an internal voting system which, for each stage, requires a threshold t of agreed results to have a valid result. Each module determines from the previous stage whether it has a valid input based on this threshold. Including the threshold voting factor, the failure rate can be described in equation 2.

$$F_M = 1 - \sum_{u=t}^n \binom{n}{u} \prod_u (1 - F_{n,m})(1 - A_{n,m}) \prod_{\forall U} (F_{n,m} A_{n,m}) \quad (2)$$

In the case of current electronic voting systems, there is a single direct line from the voter to the ballot storage device to the aggregator. Each of these systems has a failure probability, and the communication channels between them also have the ability to be compromised. This model of the current electronic voting system can be represented in Equation 2. For the entire stage to fail, there can be failures in up to t modules. For each number of failures f , we have the combination of $\binom{n}{f}$ possible failures in various modules. For each of those, we compute the probability of each number of failures f that result in a valid result.

To ensure the elimination of common potential vulnerabilities, the source code for each system will be passed through a commonality checker such as PLAG or SMAT [25][29]. This system tests for similarities between code, and is commonly used to detect cheating in assignments. In addition to the source code, it is prudent to examine the compiler used and in fact, varying the actual compilers used aids in preventing the external introduction of common vulnerabilities.

The use of common components on the SAVE system can be modeled by starting with the threshold failure model and collapsing the number of (effectively) independent modules. Then, the threshold equation reduces the effective number of components, resulting in an strictly higher probability of failure than any system with full diversity.

Adding the communication channel into the mix adds another product factor into each possible failure rate but does not fundamentally change the equation. Using a common communication channel (the most likely scenario) between components (e.g. ethernet) simply adds a common term shown in Equation 3.

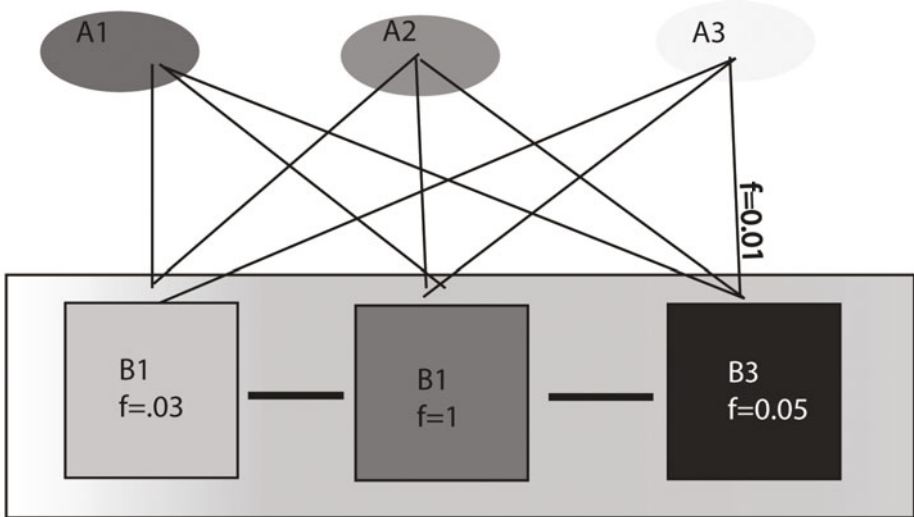


Fig. 2. Example of a stage in the SAVE system. Each module receives input from each module in the previous stage. The inputs each have a probability of being corrupted, and each module has the possibility of being corrupted. The middle module in this example was written by a nefarious programmer and thus is always compromised. The other two modules prevent the compromised module from compromising the entire election.

This factor accounts for the need to collect the ballots from the previous stage modules.

$$F_M = 1 - (1 - f_{comm}) \sum_{u=t}^n \binom{n}{u} \prod_u (1 - F_{n,m})(1 - A_{n,m}) \prod_{\forall U} (F_{n,m} A_{n,m}) \quad (3)$$

With storage (caching) of intermediate ballots, SAVE modules can batch up and transmit ballots at a later time if the network connection fails or is intentionally disrupted.

6 Discussion

6.1 Cost Considerations

The primary objection to the N-version programming SAVE in general is the additional cost of building independent modules, platforms, and their respective certification. This consideration is dealt with in SAVE by the specification of small modules and the communication protocol such that modules are small, easy to understand, and less able to obfuscate faulty or malicious code. The platforms (the computer, operating system, and possible libraries/environments such as Java) may already be certified and examined. Code can be made available online for review by anyone at minimal cost.

Given that an entire top-to-bottom system was written by Soyini Libud at MIT [18] in less than a year, it is clear that companies and interest groups with modest budgets [17] (less than the cost of a full page ad in the New York Times) can write a SAVE module. Furthermore, instead of having large companies that build end-to-end proprietary voting systems, a wide community of developers could flourish.

6.2 Improved Security

One of the most vulnerable parts of any system is the communication channel. In SAVE, messages are passed in encrypted versions of plaintext XML. This arrangement produces messages that often have the same format and even content. Depending on the security of the cryptography implementations, these repetitious messages might harm security.

The cryptographic keys could also be compromised by election officials. One method of dealing with this problem is issuing distributed keys, in which each individual does not have the entire key, and it takes a number of officials to collude in order to compromise the election. However, the extent to which the official would need to collude would also preclude a valid result in any other election scheme.

6.3 Transparency

One of the principles governing security of voting is the privacy of the ballot. A vote should not be traced back to the voter. This property is important as a defense against coercion, bribery and threats against a voter. One of the benefits of using cryptographic security is the ability to provide encrypted receipts to

voters [12]. These receipts cannot prove what the voter voted on, but they can prove that the vote is included in the tally. These receipts can help inspire confidence in the entire process, showing voters that their vote was included in the final tally.

In addition, the inclusion of witness modules written by political parties and interest groups enables voters to see that votes were cast during valid voting periods and helps reduce post-voting litigation due to accusations of 'stuffing the ballot box.'

7 Conclusions

Paper balloting presents myriad opportunities for defrauding the election process, and VVPT that have a legal preference over electronic systems retain the same possibility for manipulation. VVPT systems also present the opportunity to sow confusion amongst election officials and voters as to which result is actually valid.

Three different versions of SAVE systems have been written by students at MIT. These systems have been implemented as C++ and Java, and coded independently.

Whereas paper has no cryptographic security, electronic systems enable a better way of ensuring the validity and privacy of ballots. By creating systems with n-version programming, failures from denial of service attacks, nefarious vendors and malicious poll workers can be mitigated.

The SAVE system takes the existing reliability of electronic voting systems and adds resilience to both internal failures and malicious attack. The greater the diversity achieved in the modules, the greater the protection from failure. As always, cost is a factor, but as a modular architecture, SAVE systems should be easy to build, understand and validate.

References

1. Albert, R., Albert, I., Nakarado, G.L.: Structural vulnerability of the North American power grid. *Physical Review E* 69(2), 25103 (2004)
2. Ansolabehere, S.: Voting Machines, Race, and Equal Protection. *Election Law Journal* 1(1), 61–70 (2002)
3. Ansolabehere, S., Stewart, C.S.: Residual Votes Attributable to Technology. *The Journal of Politics* 67(2), 365–389 (2005)
4. Bailey, M., Cooke, E., Jahanian, F., Watson, D., Nazario, J.: The Blaster Worm: Then and Now. *IEEE Security & Privacy* 3(4), 26–31 (2005)
5. Bannet, J., Price, D.W., Rudys, A., Singer, J., Wallach, D.S.: Hack-a-vote: Security issues with electronic voting systems. *IEEE Security & Privacy Magazine* 2(1), 32–37 (2004)
6. Bederson, B.B., Lee, B., Sherman, R.M., Herrnson, P.S., Niemi, R.G.: Electronic voting system usability issues. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 145–152 (2003)
7. Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H.F., Thatte, S., Winer, D.: Simple Object Access Protocol (SOAP) 1.1 (2000)
8. Calandrino, J.A., Feldman, A.J., Halderman, J.A., Wagner, D., Yu, H., Zeller, W.P.: Source Code Review of the Diebold Voting System

9. MIT CalTech: Voting Technology Project. Voting: What is, what could be (2001)
10. Chaum, D.: Blind signatures for untraceable payments. In: *Advances in Cryptology: Proceedings of Crypto*, vol. 82, pp. 23–25 (1982)
11. Chaum, D.: Secret-ballot receipts: True voter-verifiable elections. *IEEE Security & Privacy Magazine* 2(1), 38–47 (2004)
12. Chaum, D., Ryan, P.Y.A., Schneider, S.: A Practical Voter-Verifiable Election Scheme. In: *Proceedings of Computer Security-Esorics 2005: 10th European Symposium on Research in Computer Security*, Milan, Italy, September 12-14 (2005)
13. Feldman, A.J., Halderman, J.A., Felten, E.W.: Security Analysis of the Diebold AccuVote-TS Voting Machine. Center for Information Technology Policy. Princeton University, Princeton (September 13, 2006)
14. Fujioka, A., Okamoto, T., Ohta, K.: A practical secret voting scheme for large scale elections. In: Zheng, Y., Seberry, J. (eds.) *AUSCRYPT 1992*. LNCS, vol. 718, pp. 244–251. Springer, Heidelberg (1993)
15. Jefferson, D., Rubin, A.D., Simons, B., Wagner, D.A.: A Security Analysis of the Secure Electronic Registration and Voting Experiment, SERVE (2004)
16. Jones, D.W.: Problems with Voting Systems and the Applicable Standards (2001)
17. Kovalev, I.V., Grosspietsch, K.E.: An Approach for the Reliability Optimization of N-Version Software under Resource and Cost/Timing Constraints. Institut für Autonome intelligente Systeme (1999)
18. Liburd, S.S.D.: An N-version electronic voting system (2004)
19. Mercuri, R.: A better ballot box? *IEEE Spectrum* 39(10), 46–50 (2002)
20. Mitrou, L., Gritzalis, D., Katsikas, S.: Revisiting legal and regulatory requirements for secure e-voting. In: el Hadidi, M., et al. (eds.) *Proc. of the 16th International Information Society Conference (IFIP/SEC-2002)*, Egypt, pp. 6–8 (2002)
21. Mitrou, L., Gritzalis, D., Katsikas, S., Quirchmayr, G.: Electronic Voting: Constitutional And Legal Requirements, And Their Technical Implications. In: *Secure Electronic Voting* (2003)
22. Moynihan, D.P.: Building Secure Elections: E-Voting, Security, and Systems Theory. *Public Administration Review* 64(5), 515–528 (2004)
23. Nickerson, D.W., Friedrichs, R.D., King, D.C.: Partisan Mobilization Campaigns in the Field: Results from a Statewide Turnout Experiment in Michigan. *Political Research Quarterly* 59(1), 85 (2006)
24. Penha-Lopes, J.M.: Why use an open source e-voting system? In: *Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education*, pp. 412–412 (2005)
25. Plag, I.: *Morphological Productivity: Structural Constraints in English Derivation*. Walter de Gruyter, Berlin (1999)
26. Pointcheval, D.: Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology* 13(3), 361–396 (2000)
27. Rubin, A.: Security Considerations for Remote Electronic Voting over the Internet. *Communications Policy and Information Technology: Promises, Problems, Prospects* (2002)
28. Xenakis, A., Macintosh, A.: Procedural security in electronic voting. In: *Proceedings of the 37th Annual Hawaii International Conference on System Sciences*, pp. 116–123 (2004)
29. Yamamoto, T., Matsushita, M., Kamiya, T., Inoue, K.: Measuring Similarity of Large Software Systems Based on Source Code Correspondence. In: Bomarius, F., Komi-Sirviö, S. (eds.) *PROFES 2005*. LNCS, vol. 3547, pp. 530–544. Springer, Heidelberg (2005)

A Modular Voting Architecture (“Frog Voting”)

Shuki Bruck¹, David Jefferson², and Ronald L. Rivest³

¹ CalTech

bruck@caltech.edu

² Lawrence Livermore Laboratory

d_jefferson@yahoo.com

³ Computer Science and Artificial Intelligence Laboratory

MIT, Cambridge, MA 02139

rivest@mit.edu

Abstract. This paper presents a new framework—a reference architecture—for voting that we feel has many attractive features. It is not a machine design, but rather a framework that will stimulate innovation and design. It is potentially the standard architecture for all future voting equipment. The ideas expressed here are subject to improvement and further research.

(An early version of this paper appeared in [2, Part III]. This version of the paper is very similar, but contains a postscript (Section 8) providing commentary and discussion of perspectives on this proposal generated during the intervening years between 2001 and 2008.)

1 A Modular Voting Architecture—Overview

We call our framework A Modular Voting Architecture (AMVA). With AMVA votes are recorded on physical items we call “Frogs”—a term chosen specifically to convey no information about the physical form of the recording device. (Frog is not an acronym. A picture of a Frog was chosen as a convenient piece of clip art designed to get the reader’s mind off of a specific technology, such as paper, mechanical devices, computer screens, or voice recorders.) A Frog is more than a ballot because it contains information besides the list of votes cast. It also contains information about the official who signed in the voter, about the precinct, and about the form of the ballot. A Frog should be a physical object. It is deposited and becomes part of the audit trail when the voter casts her vote [1].

A central design choice for this architecture is that we separate the processes of (1) recording a voter’s choices on a Frog (capture of voter’s selections), and (2) casting the vote using the Frog as input. This separation is familiar to voters using paper ballots or optical scan equipment, but not to those who use typical DRE (Direct Recording Electronic) machines.

This separation is crucial. It can help reduce or even eliminate a number of problems with existing voting technology (as discussed in [2]). These problems include security threats posed by complex electronic voting machines, the decline

¹ For convenience in this paper, voters will be feminine.

in openness and public control, the need for improved ballot designs, the need for more voter feedback so voters can catch errors, and obstacles to creating independent audit trails, especially on electronic machinery.

The current voting process consists of several distinct steps:

First, voters sign in. Three important things happen when voters sign in. They state who they are. They may be asked for identification (authentication). And they are given an initialized and official ballot that contains the races for which the voter is eligible to vote, based on the voter's residence and/or party affiliations.

Second, there is a mechanism to capture voters' selections—for example, a paper ballot or a panel of levers or buttons. The ballot presents choices to the voter and the voter selects the preferred alternatives. We call this *vote capture*.

Third, voters *confirm* their selections.

Fourth, votes are *cast*. This is the critical moment for the security of the ballot. Literally, the voter relinquishes control of the vote, and gives it over to the vote management system.

Fifth, votes are *counted*.

Sixth, votes are *audited* (by recounting a statistical sample of the cast ballots).

Many systems combine steps two, three, and four. We think that both security and innovation suffer as a consequence.

Security suffers because too much is required of a single, increasingly complex machine.

Design and innovation suffer because the process for certifying equipment is “all-at-once”—innovation in one aspect can't be done without re-certifying the entire system. The design of vote-capture components and user interfaces should evolve quickly, without being tied to certification of other parts.

At the same time, we do need strict standards for security of the casting device and reliability of counting mechanisms. Putting everything in one box significantly limits the ability to have the best vote-capture components while achieving a high level of security.

AMVA captures what we consider to be the strengths of both the optical scanning and direct recording electronic systems.

Though optical scan is perhaps today's dominant voting technology, it has its own problems, including the high cost of printing ballots, the inflexibility of the user interface, and the inaccuracy of the scanners. A good feature of optical scan is that the ballot is directly filled out by the voter and becomes part of the audit trail.

Electronic DRE machines (without VVPAT's) have no printing costs and offer flexible user interfaces. When issues such as rotating candidate positions on the ballot and supporting multiple languages on a ballot are considered, it seems clear that some form of electronic vote entry may someday become the dominant voting technology. Furthermore, the cost of all forms of electronic equipment continues to drop rapidly; a machine costing \$5,000 today might cost \$500 in a decade.



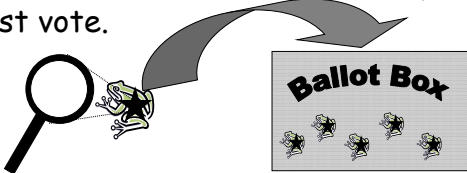
1. *Initialization:* Voter signs in, gets Frog 
 2. *Vote-Capture:* Voter records choices ★ on Frog, using vote-capture device. 
 3. *Vote-Casting:* Voter uses vote-casting device to confirm selections made, and to cast vote.
- 

Fig. 1. A Modular Voting Architecture (“Frog Voting”)

However, electronic voting systems are likely to be complex, and complexity is the enemy of security. Such voting systems are likely to be software-based. Ensuring that software is bug-free and secure is notoriously difficult. There may be little that an election official can do beyond accepting a vendor’s “trust us” statement, an unacceptable situation, or trusting an equipment certification process that does not include a rigorous security evaluation.

By separating vote capture from vote casting, and having the voter transport her ballot on a Frog from one operation to the other, we achieve several security-related objectives.

First, the voter’s ballot is recorded on a physical object (the Frog) that becomes part of the audit trail once the vote is cast.

Second, the certification of a vote-capture device may have different standards than that of a vote-casting device. The vote-capture device might have lots of graphics-oriented software that is difficult to certify, while the more critical vote-casting device could be exceptionally simple and easily certifiable.

Third, different manufacturers could produce the vote-capture equipment and the vote-casting equipment. (The recording formats and interfaces for Frogs would be standardized and public.) The ability to replace any component with a similar component from a different manufacturer (e.g., for a recount) can assist in reducing the likelihood that corrupt vendor employees could bias an election.

We imagine that election officials purchase Frogs in bulk in blank, uninitialized form. Thus, Frogs may be considerably cheaper than printed paper or optical scan ballots. A blank Frog may be a blank piece of paper, a blank memory card costing twenty cents or less, or some other medium with suitable properties. Some form of electronic memory might eventually become the preferred representation of a Frog.

Roughly, voting with a Frog works as follows:

First, when a voter arrives at a poll site to vote, she identifies herself (and authenticates herself as necessary) to an election official. The election official takes a blank Frog, “initializes” it, and gives it to the voter. Alternatively, the voter arrives with a Frog.

Second, the voter places her Frog in the appropriate “vote-capture” equipment and makes her choices, which are recorded on the Frog.

Third, the voter then takes her Frog from the vote-capture equipment to the “vote-casting” equipment, confirms her selections, and then casts her vote. Her Frog is “taken hostage” and retained as part of the audit trail.

These steps should take place privately, so that the voter’s vote cannot be observed.

2 Frog Initialization

Initializing a Frog records on the Frog the identity of the authorizing election official. It also specifies the election and precinct, the corresponding ballot style (that is, which races and candidates are to be presented to the voter), the language to use, and what candidate rotation parameters (if any) are to be used. The identity of the voter is not recorded.

We imagine that the election official has a small device for initializing Frogs as necessary. Each election official may have a unique “key” that must be inserted in order to operate the device, which specifies the official’s identity, and which counts the number of Frogs initialized by each official that utilizes that device.

In short, initializing a Frog is similar to having ballots “printed on demand.”

3 Vote Capture

When a voter puts an initialized Frog into the vote entry equipment, it presents the voter with the appropriate ballot choices, and allows the voter to enter her selections. The voter is given generous feedback at all stages, and may change her vote easily.

In a paper-based system, the Frog may be a scannable paper ballot. Marking the paper ballot is the vote-capture stage.

In an electronic system, the vote-capture stage consists of a session at an electronic screen or with a personal computer (PC). When the voter is satisfied with her choices, she pushes a “vote-entry finished” button that causes the voter’s choices to be recorded on the Frog. The voter removes the Frog so that she may place it in the vote-casting equipment.

4 Vote Casting

The vote-casting equipment has five functions when the voter casts her vote. The first is *vote-confirmation*. The Frog is “read” (scanned, electronically read,

or whatever is appropriate for this form of Frog), and the voter’s choices are displayed to the voter. The voter is asked to confirm that these are indeed her choices. If they are not, the voter’s Frog is returned to her unaltered so that she may return to the vote-entry station.

The second function is *vote signing*. The Frog is digitally signed—a cryptographic digital signature of the voter’s choices is made by the vote-casting equipment and entered into the Frog. The digital signature key is unique to that vote-signing equipment. It identifies the machine being used and authenticates the vote as having come from that machine. Different machines use different keys. The signature does not identify the voter in any way.

The third function is *vote copying*. The equipment makes an electronic digital copy of the signed vote. This copy will be communicated later on to the recording system.

The fourth function is *vote sealing*. The Frog is “sealed” or frozen so that no further changes may be made to the information it contains. With an electronic memory card Frog, a fuse might be blown that disables further writing. With paper, sealing might be more difficult to do and might have to be omitted, although laminating the ballot might serve the same purpose.

The fifth function is *Frog capture*. The Frog is taken hostage and saved as part of the audit trail.

5 Vote Recording

When the election is closed, the vote-casting equipment transmits the electronic copies of the votes, including initialization data and digital signature, to the recording system. Each vote-casting machine displays the number of votes it has signed and transmitted, which is recorded by the election officials. The Frog-initialization machines also display the number of Frogs they have initialized; these numbers are also recorded.

The recording system makes all votes and associated counts publicly available. The votes might, for example, be posted on the Web. Anyone can check the consistency of the counts, verify the digital signatures on the votes, and add up the totals to see who has won each race. We believe that this form of “universal verifiability” greatly enhances security and improves confidence in the result. Universal verifiability of all votes is possible today on all systems except lever machines and several models of DREs. Until recently, Los Angeles County, California created an electronic copy of all ballots cast—the actual image of the punch cards. The ballots could be publicly inspected.

6 Specific Examples of Frogs

The separation between vote capture and vote casting allows incredible flexibility in the system. Frogs can be created and cast at the polling places as is currently done. Frogs might also be created remotely and then recorded at a recording or polling place.

6.1 Paper Frogs

Hand-counted paper ballots most closely approximate the system we envision. When a voter checks in, she is provided with a blank, official ballot. The voter goes to a privacy booth and marks the ballot to correspond with her preferences (vote capture). The voter can inspect and change the ballot if needed. When the voter is satisfied with the ballot, she deposits the paper ballot in the ballot box. Some ballot boxes date, time, and precinct stamp the ballot (vote casting).

This system lacks the authorization by the election official on the ballot itself.

6.2 Electronic Frogs in Precincts

When the voter checks in, she is given a memory card, containing the appropriate information about the ballot, the precinct, and the election administrator. The card is inserted into a slot in a PC. The PC's screen then displays the alternatives, and the voter makes her choices. The machine records the choices on the memory card (vote capture). The voter then takes the memory card to a station with a simple card reading device and screen. This is a completely separate device. The screen displays the choices made by the voter. If the voter wishes to change the ballot, she takes the memory card back to the vote-capture PC. If the voter wishes to cast the ballot, she pushes the "CAST VOTE" button. The memory card is then locked and kept as a physical audit trail. The vote-casting machine records the votes electronically to be counted (vote casting).

Electronic voting today lacks a separate, physical audit trail, and the vote-capture and vote-casting stages are in a single box, which can be both less secure and more expensive.

6.3 Frogs from Anywhere

The Frog could also be a paper ballot that is printed from any computer, such as a home PC. The paper shows a list of candidates chosen, the precinct number, and other information such as the vendor's name. The paper Frog also contains a two-dimensional bar code (like in grocery stores) that contains the same information as is printed, but in a format that is readily counted. The Frog is sealed and brought to the polling place, verified, and submitted. The polling place would be equipped with Frogs and with computers for capturing votes in case the voter wanted to change the Frog prepared elsewhere.

One interesting aspect of this particular version of AMVA is, if we record the vendor name on the Frog, then vendors could be compensated on a per ballot basis. This would ensure that there was adequate money to stimulate innovation in the development of software.

7 Discussion

We imagine that each county could purchase the vote-casting equipment. It would consist of a very simple, very inexpensive box.

An independent research laboratory working under the supervision of a panel of security and voting experts would develop the specifications of the vote-casting device. These specifications would be public information, and the device could be built by anyone.

The vote-casting equipment would not be divided into “test” mode and “real” mode. The only difference between a “test” and a “real” election would be the cryptographic keys inserted into the device.

The vote-casting device does not need to understand the races being run and the candidates running for each race. The device merely displays the choices recorded on the Frog, which would be recorded and displayed in a standard text format, such as in the accompanying Figure 2. The voter would be able to scroll up and down if necessary to see everything.

```

State of Massachusetts, Middlesex County, Precinct 11
Ballot Initialized by Election Official 10
Election Closes November 7, 2004 at 8pm EST
Ballot: MA/Middlesex/1; English; No rotation
You have chosen:
  U.S. President: Mary Morris
  U.S. Vice President: Alice Applebee
  Middlesex Dog Catcher: Sam Smith (write-in)
  Proposition 1 (Casino): FOR
  Proposition 2 (Taxes): AGAINST
  Proposition 3 (Swimming Pool): FOR

```

Fig. 2. An illustration of a possible format for the information recorded on a Frog

We feel that such standardization of electronic formats for ballots will be a major step forward in the evolution of voting systems. It enables the separation of vote capture and vote casting. It provides a path towards remote voting, when and if the security of remote voting systems can be sufficiently ensured. It is both human and machine-readable, and so forms a bridge between these worlds. It enables different vendors to produce interoperable equipment for a voting system. We repeat our previous concern that systems that do not produce a separate (preferably physical) audit trail are prone to security problems.

Similarly, we feel that monolithic systems that try to incorporate everything may compromise security.

So, our design places most of the complicated user interface software in the vote-capture system, which is considered to be somewhat less “security-critical.” It does need to be reviewed, but it might be acceptable to have such a device contain proprietary code. The vote-capture system might even be run on newly purchased computers or laptops which could then be sold after the election as used equipment.

On the other hand, the security of vote-casting equipment is absolutely critical. This is the last chance for a voter to see her vote before it becomes a truly

anonymous element in the list of votes cast. The election officials and voters must have strong reason to believe that the vote-casting equipment does not, at the last instant, change the voter's vote just before it is cast.

For this reason, we feel that the vote-casting equipment should be totally "open source"—the software for such a machine should be publicly available. The procedures for ensuring that the equipment actually contains the published software should be public and followed by the election officials. Such machines should be very rigorously evaluated during certification. A county may buy several vote-casting machines for each precinct, from different manufacturers.

This division of equipment into two parts may thus solve a problem in the industry: allowing manufacturers to protect some intellectual property (the code for the vote-capture systems) while ensuring that the most security-critical portions are open-source, heavily reviewed, and highly trustworthy.

Note that the vote-casting equipment does exactly the same thing for each election: it merely displays the contents of the Frog, gets the voter's final approval, digitally signs the contents of the Frog, and makes a copy of everything. It does not need to know anything about the particular election being run (although it will use an election-specific digital signature key); the voter is herself taking responsibility for final approval. It does not even have the ability to change a user's vote, if the user does not approve it; that is the function of vote-capture. (Of course, we expect that some voters may not bother to read the final confirmation screen carefully; that is their choice. Indeed, we do not expect there are likely to be problems at this stage, although some voters may change their minds at the last instant or they may realize that they forgot to vote in some contest.)

The election officials can take the vote-casting equipment out of the closet, initialize it with the cryptographic signing key it is to use, and then power it on.

Of course, a voter should not be allowed to use the vote-casting equipment unless she has been identified as an eligible voter who has not previously voted. Some physical control of the voters at the polling place is necessary. Conceivably one could authenticate the voters at the vote-casting station, but then the issues of ballot style, language, etc. may not get handled properly, and it seems more awkward to have problems arise at this late stage if there have been problems with the voter's registration from the beginning of the process.

The use of digital signatures is an important and critical part of this design. Anyone who could forge digital signatures could forge votes. The cryptographic digital signature keys need to be carefully managed. A reasonable extension of the basic AMVA design would allow the vote-casting machinery to simultaneously use several signature modules (e.g., each on its own memory card), so that each cast vote is signed by all modules. In addition to the basic signature module supplied by an election official, there may be signature modules supplied by each political party. Requiring several signatures on a vote makes it much harder for a single individual to surreptitiously "borrow" the equipment and forge signed votes. The parties would keep a careful eye on their signature modules, not supplying them until just before the election and retrieving them as soon as the election was over.

Of course, signatures work with paper systems also. The election officer might stamp all of the relevant information on the top of the ballot. When the vote is cast, the ballot is placed in a paper sleeve that only shows the top part. The election administrator would then sign the top of the ballot without observing the votes to certify that everything about the ballot (precinct, etc.) is correct.

The voter’s anonymity is nonetheless protected. Her ballot is identified only by the name (or identification number) of the election official who authorized her to vote, and the identity of the vote-casting machine that digitally signed her vote. As long as a reasonable number of voters fall into each such category, anonymity is ensured.

8 Postscript and Discussion

This section provides some comments on this proposal based on what has happened in the years 2001–2008.

The “electronic ballot marker” (essentially the vote-capture device as described here) has appeared on the market; the Automark² is one example. Proposed federal standards for certifying electronic ballot markers are given in the Voluntary Voting System Guidelines [4]. However, the federal government is not yet proposing certifying voting system components, only complete voting systems.

This proposed federal standard also prescribes the use of digital signatures for use with voting system records, in a manner similar to what is proposed in this paper. Digital signatures do however need to be handled carefully. It should *not* be the case that a voter can be disenfranchised by malicious software that intentionally produces invalid digital signatures. Invalid digital signatures should cause an alarm to be raised for election officials to investigate, to see if there is additional evidence supporting the hypothesis that a ballot (or collection of ballots) is fraudulent. But the failure of a signature verification should probably not by itself be enough to invalidate a ballot.

The standards requisite for information interchange between voting system components is developing, most notably in the Election Markup Language (EML) [1].

The present paper could perhaps have benefitted from the useful terminology in the proposed VVSG [4], distinguishing between voting systems having *direct* verification (i.e., with the voters own eyes, as for paper ballots), and *indirect* verification (i.e. through the mediation of some device).

Another useful term introduced in [4] is that of “software independence” (see also [3]); a voting system is said to be “software independent” if it is not the case that an undetected software error can cause an undetectable change in the election outcome. The system proposed in the present paper is software independent when the ballots support direct verification; otherwise (when the vote-casting device is necessary for the voter to confirm that her choices are

² <http://www.automarkts.com/>

correctly recorded) it is not software-independent. The authors now support the proposal that voting systems should be software-independent.

The use of barcodes as suggested at one point in the present paper is problematic; if no barcodes are used then a paper ballot is directly verifiable by the voter (and the system is software-independent); otherwise the ballot is only indirectly verifiable, since a device is needed to read the barcodes (and the system is not software-independent). Barcodes may not be worth the extra complexity and potential security vulnerabilities.

There has been some interesting work on the problem of developing a very small code base for a voting system. Most notable is that of Ka-Ping Yee and colleagues [6,5], who propose using a small Python program driving a user-interface with pre-rendered graphics. Their system does both vote capture and vote casting.

References

1. OASIS. Election markup language (EML) v5.0 (2007), <http://www.oasis-open.org/specs/index.php#eml5.0>
2. CalTech/MIT Voting Technology Project. Voting: What is, what could be. Technical report (July 2001), <http://www.vote.caltech.edu/2001report.htm>
3. Rivest, R.L., Wack, J.P.: On the notion of “software independence” in voting systems, July 28 (2006), <http://vote.nist.gov/SI-in-voting.pdf>
4. TDC. Voluntary voting system guidelines, <http://www.eac.gov/vvsg>
5. Yee, K.-P.: Building Reliable Voting Machine Software. PhD thesis, U.C. Berkeley EECS Dept. (2007); Technical Report 2007-167, <http://www.eecs.berkeley.edu/Pubs/TechRpts/2007/EECS-2007-167.html>
6. Yee, K.-P., Wagner, D., Hearst, M., Bellovin, S.: Prerendered user interfaces for high-assurance electronic voting. In: USENIX/ACCURATE Electronic Voting Technology Workshop (2006), <http://zesty.ca/pubs/yee-wagner-hearst-bellovin-prui-usenix2006.pdf>

Unconditionally Secure Electronic Voting

Akira Otsuka and Hideki Imai

Research Center for Information Security (RCIS),
National Institute of Advanced Industrial Science and Technology (AIST)
1-18-13 Sotokanda, Chiyoda-ku, Tokyo 101-0021 Japan
a-otsuka@aist.go.jp, h-imai@aist.go.jp

Abstract. In this chapter, we will show how to achieve unconditional or information-theoretic security in electronic voting with the following property:

1. Even all voters and tallying authorities have unbounded computing power, the distorted integrity of the voting results can be detected and proved incorrect by every honest voter,
2. If at least one tallying authority is honest, then the privacy of the ballots are protected everlastingly even the other voters and tallying authorities are malicious and have the unbounded computing power.

We assume single trusted authority who honestly delivers a particular form of secret key to every voter and tallying authority. This authority can be destroyed before the election is started. Two information-theoretic primitives are introduced based on this pre-distributed secret key, unconditionally secure oblivious polynomial evaluation (US-OPE) and unconditionally secure publicly verifiable secret sharing (US-PVSS). These primitives make the election process unconditionally secure in the above sense and efficient. The resulting scheme requires in a case of 1 million voters, the storage complexity to store private key required for each voter is 300MB. Communication complexity to verify the whole tallying process (the heaviest part) is 27GB in a case of tolerating up to 1000 colluding users, and 220GB in a case of tolerating up to 10,000 colluders.

1 Introduction

An invention of quantum computers [20] which efficiently solves factoring problems and discrete logarithm problems may totally break most of the current public-key based information systems. More practically, TWIRL [21] proposed by Shamir and Tromer or other dedicated hardwares for factoring is threatening to change our naive estimation of secure key sizes in the near future. It is natural to hope a more secure scheme in principle which is not bothered by this everlasting game.

In this chapter, we investigate several information theoretic primitives which are useful to design more complicated distributed multiparty protocols. Oblivious Polynomial Evaluation(OPE) is one of the very useful tools where two parties, Alice and Bob, are given a polynomial and a value respectively on their private

inputs. Then, after jointly executing a protocol, Bob outputs a value of the polynomial evaluated at the value without learning any useful information about the values of their private inputs each other. OPE is first proposed by Naor and Pinkas [16] in the computational setting. In [6], they improved the efficiency of OPE, and also proposed OPE in the information theoretic setting. However, the security of their scheme depends on trustiness of a online trusted party. Our result shows that if private keys are securely distributed to each players, then unconditionally secure is efficiently implementable. The simplest implementation of our protocol requires a trusted party who engages only in the set-up phase (trusted initializer [18]). Thus no online trusted party is required.

The other primitive introduced in this paper is information theoretic version of Publicly Verifiable Secret Sharing (PVSS). PVSS is an extension of Verifiable Secret Sharing [7][11] first introduced by Stadler [22] and later improved [12][19]. PVSS is a VSS with a property that not only the players but anyone can verify that the secret is shared properly. Interestingly, PVSS is possible under the computationally bounded model or maybe under the storage bounded model, but impossible in the information theoretic model. Since the schemes based on computational assumption can utilize a public verification key to check the consistency of the shares in such a way that the adversary cannot cheat a casual user without solving a computational hard problem. However, in the information theoretic model, it is impossible since the casual user has no information theoretic advantage over the adversary. Thus, we restrict the notion of public verifiability of PVSS to that every user with his own private key can verify the integrity of PVSS. Since every user (including non-voters) is eligible to become a public verifier, this restriction will not cause a major problem in an application to e-voting.

1.1 Related Work

In electronic voting systems, the following two security properties are considered with special importance [4]:

- Secret ballots (Receipt-freeness¹)
Voter must not be able to convince others of how he or she voted.
- Unconditional integrity
Anyone should not allow incorrect tally (except with negligible probability).

Efficient schemes are ever proposed which protects one of the above property unconditionally but the other only computationally. Chaum et al. [5] proposed a scheme achieving unconditional integrity with computational receipt-freeness. Moran and Naor [15] satisfies unconditional receipt-freeness but computational integrity, and Cramer et al. [10] satisfies similar property but only unconditional secrecy of ballots. Chaum's and Moran's schemes are further designed to

¹ Secrecy of ballots is defined in many articles as primary property of electronic voting. However, receipt-freeness is strictly stronger than secrecy of ballots as receipt-freeness implies secrecy of ballots. Thus, only receipt-freeness is required here.

cope with more demanding problem of the polling booth scenario where polling machines are not trustworthy. Paper receipt from the polling machines can help voters to verify the integrity of tally via internet, while providing receipt-freeness. But generally speaking, it seems to be hard to design a scheme which satisfies both of the properties unconditionally as far as we base them on cryptographic primitives which must assume computational intractability.

Broadbent and Tapp [3] proposed electronic voting scheme which simultaneously satisfies unconditional secrecy of ballots and unconditional integrity. Their scheme is based on information-theoretic primitives primarily on cut and choose technique and multiparty computation. In their scheme, the integrity of tally totally depends on the trust of authorities, where no one outside of the tallying authorities can verify the integrity. Moreover, if one of the authorities is dishonest, then the dishonest authority can revoke any ballot from honest voters.

Our scheme simultaneously satisfies unconditional secrecy of ballots and unconditional integrity of tally. The integrity of tally is assured through information-theoretic public verifiability, where everyone with private verification key can detect the distorted integrity of the voting results and proved incorrect even if all dishonest voters and tallying authorities have unbounded computing power. If at least one tallying authority is honest, then the privacy of the ballots are protected everlastingly even the other voters and tallying authorities are malicious and have the unbounded computing power.

Achieving unconditionally secure receipt-freeness and extending our unconditionally secure scheme to polling booth scenario still remain as open problems.

2 Preliminary

In this section, we introduce the most simplified version of our information theoretic tool, hidden point evaluation technique. The setting taken in this paper is as follows. Each player is given some predistributed information described below so that each player has an information theoretic advantage over the other players. Our aim is to construct more complicated protocols like electronic voting using only this predistributed information, thus without depending on any computational assumption.

Suppose we have a prover P and verifiers V_1, \dots, V_n , all of them are probabilistic polynomial-time algorithms. Let q be a prime power, and let \mathcal{F} be a set of all univariate polynomials in $GF(q)[x]$.

For some polynomial $f_0 \in \mathcal{F}$ of degree ω chosen randomly and uniformly. The prover is given the polynomial f_0 as is. On the other hand, each verifier V_1, \dots, V_n is given a randomly and uniformly chosen hidden point (x_i, y_i) satisfying $y_i = f_0(x_i)$ for $i = 1, \dots, n$ in a way that each hidden point is only known to the corresponding verifier and that the other players including the prover have no information on which point on $y = f_0(x)$ is chosen by the verifier.

Note that in Shamir's secret sharing, the dealer(prover) knowing the polynomial also knows each share delivered to each shareholder(verifier). However, in our setting, the prover knows the polynomial but has no information on the

shares held by the shareholders. Thus, in our setting, each verifier (shareholder) also has an information theoretic advantage over the prover. Using this information theoretic advantage of the verifiers, we can verify the correctness of a polynomial posted by the prover.

Definition 1. (*Verification*) For a polynomial $f \in \mathcal{F}$ generated by the prover, the polynomial f is called **accepted** by a verifier V_i , if and only if the polynomial f satisfies $y_i = f(x_i)$.

A polynomial f is evaluated by the verifier V_i whether the polynomial f passes through a hidden point (x_i, y_i) . Obviously, there are many polynomials possibly accepted by the verifier. However, since the evaluation point is hidden, there exists no better way than a random guess for the prover to find a good polynomial $f \neq f_0$ to be accepted by the verifier. From this discussion, we have the following lemmas.

Lemma 1. (*Integrity*) For any adversary \mathcal{A} who may collapse the prover and all the verifiers except for the targeted verifier V_i , the success probability that the adversary \mathcal{A} to force the verifier V_i accept a wrong polynomial $f' \neq f_0$ is exponentially small with security parameter k .

This lemma implies that once the verifier V_i accepted a polynomial $f(x) \in \mathcal{F}$, then the polynomial is *correct*, $f(x) = f_0(x)$ for all $x \in GF(p)$, with high probability. Thus, we have the following corollary.

Corollary 1. (*Unanimity*) Suppose that a polynomial $f \in \mathcal{F}$ is accepted by a honest verifier V_i , then the probability that another honest verifier V_j to reject the same polynomial f is exponentially small.

Now we turn to the security for the prover. The polynomial f_0 given to the prover is unconditionally hidden from the verifiers colluding of up to ω verifiers.

Lemma 2. (*Secrecy*) Suppose that the prover has a polynomial $f_0 \in \mathcal{F}$ of degree ω , and that we have any adversary \mathcal{A} that have control over up to ω verifiers with hidden points $\{(x_1, y_1), \dots, (x_\omega, y_\omega)\}$. Then, the success probability for the adversary \mathcal{A} to compute $f \in \mathcal{F}$ such that f will be accepted by at least one non-colluding verifier is exponentially small.

Further, we introduce homomorphic property to the above scheme. As in many multiparty computation defined over polynomial-based secret sharing schemes, the above stated polynomial-based hidden point technique can also be equipped with homomorphic property. Observe that for any two polynomials $f_1(x), f_2(x) \in \mathcal{F}$ and two points on each polynomial $(v_i, f_1(v_i))$ and $(v_i, f_2(v_i))$, any linear combination of the two polynomials $g(x) = af_1(x) + bf_2(x)$ with a, b in $GF(q)$ satisfies the following equation:

$$g(x_i) = af_1(v_i) + bf_2(v_i).$$

More general definition follows.

Definition 2. (*Homomorphism*) For a polynomial $g(x) \in \mathcal{F}$ generated from a linear combination of polynomials $f_1(x), \dots, f_n(x) \in \mathcal{F}$, and for a commitment (a_1, \dots, a_n) generated by the prover, the commitment-polynomial pair (a_1, \dots, a_n) and $g(x)$ is called **accepted** by a verifier V_i , if and only if the following equation is satisfied:

$$g(x_i) = \sum_{j=1}^n a_j f_j(v_i).$$

3 Information Theoretic Primitives

3.1 Oblivious Polynomial Evaluation

Oblivious polynomial evaluation(OPE) is an extension of the basic primitive, oblivious transfer(OT), first introduced by Naor and Pinkas [16]. OPE is a two party protocol where Alice is given a polynomial $f(x)$ on her private input, and Bob is given a value x_0 on his private input. After executing a protocol, Bob outputs a value $y_0 = f(x_0)$ (with negligible error probability) in a way that Alice has no information (or learns negligible amount of information) on the Bob's input x_0 and that Bob has no more information (or learns negligible information) on the Alice's private input $f(x)$ than that can be implied from y_0 .

Definitions and Bounds. In [13], OPE is formalized in the information theoretic setting. We restate the definitions and bounds on US-OPE in the following.

Definition 3. (*ϵ -correct OPE*) A OPE protocol π is called **ϵ -correct** if after executing the protocol π with honest players, there exists ϵ satisfying the following equation:

$$\Pr(y \neq y_0 : (\perp, y) \leftarrow \pi(f, x_0)) \leq \epsilon$$

where y_0 is the correct output such that $y_0 = f(x_0)$.

Definition 4. (*ϵ -private OPE*) Let F , X and Y be the random variables representing the polynomial f on Alice's private input, the value x_0 on Bob's private input, and y on Bob's private output. A OPE protocol π is called **ϵ -private** for Bob if for any possible behavior of Alice,

$$I(\text{View}_A; X) \leq \epsilon$$

where $I(\cdot; \cdot)$ is Shannon's mutual information, View_A is a random variable which represents Alice's view after completion of the protocol π , X is a random variable representing Bob's input x_0 .

Similarly, an OPE protocol π is called **ϵ -private** for Alice if for any possible behavior of Bob, there exists ϵ such that

$$I(F; X) \leq \epsilon,$$

$$I(F; \text{View}_B | XY) \leq \epsilon.$$

where $View_B$ is a random variable which represents Bob's view after completion of the protocol π , Y is a random variable representing Bob's output y_0 .

An OPE protocol π is said to be ϵ -private if it is ϵ -private for Alice and Bob. In the special case of $\epsilon = 0$, we call the protocol π is *perfectly private*.

Let K_A and K_B be random variables representing information held by Alice and Bob respectively before initiating the OPE protocol. The following theorem gives the lower bound on the initial information.

Theorem 1. (*Lower Bounds on Private Keys*)

If a OPE protocol π is perfectly private, then π satisfies the following bounds.

$$H(K_A) \geq H(F), H(K_B) \geq H(X) + H(Y|X)$$

Proofs are given in [13].

Construction. Now we will give the optimal construction of perfectly private OPE.

Protocol OPE

Initial Information: Private Keys

Alice's key: $R(x) \in GF(q)[x]$ of degree at most n ,

Bob's key: (d, R_d) where $d \in GF(q)$ and $R_d = R(d)$.

OPE Phase

Alice's input: $f(x) \in GF(q)[x]$, $\deg f(x) \leq n$,

Bob's input: $x_0 \in GF(q)$.

1. Bob sends to Alice $e = x_0 - d$,
2. Alice sends to Bob $g(x) = f(x + e) + R(x)$,
3. Bob outputs $y = g(d) - R_d$.

Theorem 2. *The above stated protocol is a perfectly-correct and perfectly-private oblivious polynomial evaluation. Moreover, it is optimal regarding its private key size.*

Proof. Correctness is obvious. Since if Alice and Bob are both honest, then after the completion of the above protocol, Bob outputs the correct value $f(x_0)$ with probability 1 (perfectly correct). To prove privacy for Bob, note that d is uniformly distributed and not known to Alice, thus $H(X|K_A View_A) = H(X)$ holds. Privacy for Alice follows from the fact that every action of Bob's amounts to choosing an x_0 . However, given x_0 and $f(x_0)$, he can evidently simulate his view of an execution of the above protocol: he simply chooses randomly d and R_d and polynomial $g(x)$ such that $g(d) = f(x_0) + R_d$. Since this uses no further knowledge of f , the security condition $H(F|K_B View_B) \leq H(F|XY|K_B View_B) = H(F|XY)$ holds.

Size of the private keys clearly meets the lower bound in Theorem 1 assuming uniform distribution over all inputs.

3.2 Publicly Verifiable Secret Sharing

We will introduce an information theoretic version of the powerful and important primitive, publicly verifiable secret sharing (PVSS). PVSS is first introduced by Stadler [22]. PVSS is a variant of verifiable secret sharing schemes with additional property that every casual user can verify the consistency of publicly posted encrypted shares. This property enables electronic voting schemes to enjoy the important property that every citizen can check the correctness of voting and tallying process.

PVSS schemes [22,19] based on computational assumption allows any casual users can become a public verifier. The schemes uses public verification key so that every casual user can obtain the key and verify the encrypted shares posted on the bulletin board. This is a very nice feature of PVSS.

In the information theoretic PVSS (US-PVSS), verification of the shares with a single public verification key is impossible. Thus, even public verifier must be delivered a private verification key. In the single public verification key setting, public verifier does not have any information theoretic advantage, that is, it is always possible for the adversary (with unbounded computing power) to cheat the public verifiers with invalid shares.

Definitions

Definition 5. A US-PVSS consists of a dealer, N participants P_1, \dots, P_N such that each has a private encryption function E_i and a private decryption function D_i shared with the dealer, public verifiers with a private verification key, a monotone access structure $\mathcal{A} \subseteq 2^{\{1, \dots, N\}}$, and algorithms *Share*, *Recover*, and *PubVerify* which operate as follows:

- *Share*: The dealer uses public encryption function to distribute the shares by calculating $S_i = E_i(s_i)$ for $1 \leq i \leq N$. The dealer then publishes each share S_i .
- *Recover*: If a group of participants want to recover the secret, they run *Recover*, which has the property that $\forall A \in \mathcal{A}$ it is infeasible to calculate s from $\{S_i \mid i \in A\}$.
- *PubVerify* : To verify the validity of all encrypted shares, *PubVerify* is run by any inquiring party with private verification key v_k . This algorithm has the property that

$$\begin{aligned} \exists u \forall A \in \mathcal{A}, \text{ PubVerify}(v_k, \{S_i \mid i \in A\}) &= \text{accept} \\ \Rightarrow \text{Recover}(\{D_i(S_i) \mid i \in A\}) &= u \wedge \Pr[u \neq s] \leq \epsilon. \end{aligned}$$

Any participants and the dealer can be a public verifier (we call them simply Verifiers in the following) if private verification key is provided.

Definition 6. A protocol π is ϵ -secure US-PVSS if it satisfies the following three properties.

1. *Completeness:*

A PVSS is said to be complete if whenever the dealer is honest (and the (unique) value for s is recoverable by the participant(s)), the verifier accepts the proof as valid with the error probability equal to or less than ϵ . It is said to be perfect if $\epsilon = 0$.

2. *Soundness:*

A PVSS is said to be sound if whenever the unique s is not recoverable, the verifier accepts the proof with probability equal to or less than ϵ . It is said to be perfect if $\epsilon = 0$.

3. *Secrecy:*

A PVSS is said to be secret if any group not in the access structure can not retrieve s . The probability gain against the secret s is equal to or less than ϵ . It is said to be perfect if $\epsilon = 0$.

Construction. Our construction of US-PVSS is a combination of the hidden point evaluation technique described in the Preliminary section and the US-OPE technique introduced in the previous section. The main idea is the following. Each VSS share in our scheme is described as a polynomial. This share polynomial is a linear combination of polynomials pre-distributed as Dealer's VSS-key, and it is verifiable with private verification keys using hidden point evaluation technique. Furthermore, the share polynomial is encrypted using US-OPE. Thus, the original share is encrypted using the secrecy property of US-OPE. On the other hand, the encrypted share is still verifiable since US-OPE obviously leaks one point (this point is designed to be equal to the private verification key) on the original share polynomial.

Protocol US-PVSS**Initial Information: Private Keys**

Dealer

$$\begin{cases} \text{VSS-key} & F_1, F_2 \in GF(q)[x, y] \text{ of degree } T \text{ and } t \\ \text{OPE-key} & R_j \in GF(q)[x] \text{ of degree } 2T \quad (1 \leq j \leq N) \end{cases}$$

Player_j

$$\{\text{OPE-key} \quad R_j \in GF(q)[x] \text{ of degree } 2T \quad (1 \leq j \leq N)\}$$

Verifier_k

$$\begin{cases} \text{VSS v-key} & v_k \in GF(q) \\ & F_1(v_k, y), F_2(v_k, y) \in GF(q)[y] \\ \text{OPE p-key} & v_k, R_j(v_k) \in GF(q) \quad (1 \leq j \leq L) \end{cases}$$

PVSS Phase

Dealer's input: secret s

Share: Dealer first chooses α depending on s such that $s = F_1(0, 0) + \alpha F_2(0, 0)$. A share for Player j is computed as $s_j(x) = F_1(x, j) + \alpha F_2(x, j) + R_j(x)$. Then, Dealer publishes the commitment α and an encrypted share $s_j(x)$.

Recover: Let $A \in \mathcal{A}$ be the set of players trying to recover a secret. Now they have a set of encrypted shares $\{s_j(x) \mid j \in A\}$. To recover a secret, simply compute the interpolate the secret from the decrypted shares $\{s_j(0) - R_j(0) \mid j \in A\}$.

PubVerify : Verifier $_k$ will accept (or reject) the encrypted share $s_j(x)$ with the commitment α if the following conditions satisfied:

$$s_j(x)|_{x=v_k} = F_1(v_k, y)|_{y=j} + \alpha F_2(v_k, y)|_{y=j} + R_j(v_k)$$

Theorem 3. *The above protocol is a US-PVSS satisfying perfect-completeness, ϵ -soundness and perfect-secrecy. Moreover, if the above protocol is constructed over $GF(q)$ and the number of public verifiers is upper-bounded by L , then the success probability for all adversary to break the soundness property is at most L/q .*

Proof. Completeness is obvious. Since if the dealer is honest, all honest Verifier $_k$ accept all encrypted shares in *PubVerify* with probability 1.

To prove soundness, let $A, B \in \mathcal{A}$ be the set of players which outputs different value: $Recover(\{D_i(S_i) \mid i \in A\}) \neq Recover(\{D_i(S_i) \mid i \in B\})$. Then there exists at least 1 share S_i where $i \in A \cup B$ such that S_i is invalid, thus $S_i \neq F_1(x, i) + \alpha F_2(x, i) + R_i(x)$, and there exists at least 1 honest verifier $k \in \{1, \dots, L\}$ who accepts the invalid encrypted share S_i . From integrity (Lemma [II](#)) and unanimity (Corollary [II](#)), the probability that this situation happen is less than L/q . This probability is exponentially small with the security parameter $|q|$.

Secrecy is also trivial from the secrecy property of the underlying Shamir's polynomial-based secret sharing scheme and the secrecy property of US-OPE.

4 Unconditionally Secure Electronic Voting

4.1 Model

We follow the bulletin board model for electronic voting as introduced by Benaloh et al. [\[8,2\]](#). The model assumes public bulletin board with which every player can post their message to it. Players are comprised of a set of tallying authorities, a set of voters Voter, and a set of passive public verifiers. An election proceeds in two phases. The first phase is the voting phase. In this phase, each voter posts his ballot to the bulletin board. Each ballot consists of encrypted shares of his vote, its commitment to prove the consistency of the shares and a proof that the ballot contains 0 or 1 in the two-value vote. Since the voters need not be anonymous in this scheme, it is trivial to prevent double voting. Only valid ballots will be accepted. The second phase is the tallying phase. In this phase, tallying authorities are involved. They will check each ballot posted on the bulletin board. Then, they decrypt and sum up the shares, like multiparty computation, and post each sum! of the shares.

The property required to voting schemes is informally stated as follows.

- Eligibility
Ensures every eligible voter posts at most one ballot.
- Privacy
Ensures the secrecy of the contents of ballots.

– Integrity

Ensures that any party, including public verifiers, can be convinced that all valid votes have been included in the final tally.

More formally an information-theoretically secure electronic voting is defined as follows.

Definition 7. (*efficient electronic voting scheme*) Suppose we have three kinds of players, a finite set of voters, Voter_i ($i = 1, \dots, M$), a finite set of tallying authority, Authority_j ($j = 1, \dots, N$), and a finite set of passive public verifiers, Verifier_k ($k = 1, \dots, L$). Each player has its own private information X_i, Y_j, Z_k respectively. Further, let P be public information and R_i ($i = 1, \dots, M$) be internal random coins of each voter. In an efficient electronic voting scheme, there exists the following three phases:

1. Voting Phase:

Given a private information X_i , public information P and internal random coin R_i , a voter Voter_i decides his vote $s_i \in \{0, 1\}$ and computes and writes information E_{ij} ($j = 1, \dots, N$) on a bulletin board.

2. Tallying Phase:

Given a private information Y_j , public information P and information on bulletin board $\{E_{ij}\}$ ($i = 1, \dots, M$), an tallying authority Authority_j outputs S_j .

3. Verification Phase:

Given a private information Z_k , public information P and information on bulletin board $\{E_{ij}\}$, a public verifier, Verifier_k , outputs accept or reject on every E_{ij} for $i = 1, \dots, M$ and $j = 1, \dots, N$. Furthermore, given a private information Z_k , public information P and the outputs of tallying authorities, $\{S_j\}$ ($j = 1, \dots, N$), a public verifier, Verifier_k , outputs the final tally S or \perp .

Definition 8. (ϵ -secure electronic voting) An electronic voting scheme is called ϵ -secure in information theoretical sense if it satisfies the following properties.

1. Eligibility:

There exists a function f which maps E_{ij} , the information on the bulletin board, to a single voter Voter_i such that for all $i_1 \neq i_2$, where $i_1, i_2 \in \{1, \dots, M\}$, and for all $j \in \{1, \dots, N\}$, the following is satisfied:

$$f(E_{i_1j}) \neq f(E_{i_2j}).$$

2. Privacy:

Let \mathbf{Y}_t be the collusion of tallying authorities with t authorities, and let \mathbf{Z}_T be the collusion of public verifiers with T verifiers. Let \mathbf{P} be the public information including the information written on the bulletin board, hence $P, \{E_{ij}\}, \{S_j\}$ and S . Then, given $\mathbf{Y}_t, \mathbf{Z}_T, \mathbf{P}$, for every algorithm \mathcal{A} , for every i , for every choice of t corrupting tallying authorities and for every choice of T corrupting public verifiers, the success probability of \mathcal{A} to guess the value of the vote s_i of Voter_i over random guess is less than ϵ . That is,

$$\left| \Pr[\mathcal{A}(\mathbf{Y}_t, \mathbf{Z}_T, \mathbf{P}) = s_i] - \frac{1}{2} \right| \leq \epsilon.$$

3. Integrity:

Let Verifier_{k_1} and Verifier_{k_2} be two public verifiers with private information Z_{k_1} and Z_{k_2} respectively. Given all information on the Bulletin Board $\{E_{ij}\}$ and given the outputs of all tallying authorities $\{S_j\}$, for every choice of k_1, k_2 , the probability that Verifier_{k_1} and Verifier_{k_2} outputs the different final tally is less than ϵ . That is, for every $k_1, k_2 \in \{1, \dots, L\}$, there exists $\epsilon > 0$ such that

$$\Pr[S' \neq S''; S' \leftarrow \text{Verifier}_{k_1}(\{E_{ij}\}, \{S_j\}), S'' \leftarrow \text{Verifier}_{k_2}(\{E_{ij}\}, \{S_j\})] \leq \epsilon,$$

where S' and S'' are outputs of Verifier_{k_1} and Verifier_{k_2} as final tally. Note that $S', S'' \in \mathbf{Z} \cup \{\perp\}$.

Further, if there exists a Verifier_k who output a final tally S , then there exists a list of voters $V = \{i_1, i_2, \dots, i_{M'}\} \subseteq \{1, \dots, M\}$ whose casted votes are valid and hence counted, and then S satisfies the following relation:

$$S = \sum_{i \in V} s_i$$

where s_i is the value of Voter $_i$'s vote.

For the privacy property of electronic voting schemes, one may notice that the collusion of voters is not considered in the definition. This is mainly because, in the extreme case, it is trivial that every voter except for a targeted voter is corrupted, then the content of the targeted voter's vote is revealed from the final tally and the choice of corrupted voters' votes. In the other words, the level of privacy highly depends on the number of honest voters. Thus, we excluded the corruption of voters from the privacy definition. The construction proposed later in this paper is actually robust against the corruption of fairly large portion of voters only if there exists enough number of honest voters.

4.2 Parameters

In the following, we will use the parameters listed below.

- L : number of public verifiers
- M : number of eligible voters
- m : number of participating voters ($m \leq M$)
- N : number of authorities
- T : maximum number of malicious verifiers
- t : maximum number of malicious authorities

4.3 Construction

A construction of electronic voting scheme based on bulletin board model [8, 2] is given in the information theoretic model. Our construction is based on US-OPE and US-PVSS described in the previous section. The construction is separated 4 phases: (1) description of private keys, (2) Voting Phase, (3) Verification Phase, (4) Tallying Phase.

Initial Information: Private KeysVoter_{*i*}

$$\begin{cases} \text{VSS-key} & S_{i1}, S_{i2} \in GF(q)[x, y] \text{ of degree } T \text{ and } t \\ \text{OPE-key} & R_{ij} \in GF(q)[x] \text{ of degree } 2T \quad (0 \leq j \leq N) \end{cases}$$

Authority_{*j*}

$$\{\text{OPE-key} \quad R_{ij} \in GF(q)[x] \text{ of degree } 2T \quad (1 \leq i \leq M)$$

Verifier_{*k*}

$$\begin{cases} \text{VSS v-key} & v_k \in GF(q) \\ & S_{i1}(v_k, y), S_{i2}(v_k, y) \in GF(q)[y] \\ & \text{of degree } t \quad (1 \leq i \leq M) \\ \text{OPE h-key} & v_k, R_{ij}(v_k) \in GF(q) \\ & (1 \leq i \leq M, 0 \leq j \leq N) \end{cases}$$

All private keys are chosen randomly and uniformly.

Every tallying authority in our scheme is given a private key as Verifier:
 $\{\text{Verifier}\} = \{\text{Voter}\} \cup \{\text{Authority}\} \cup \{\text{Public Verifier}\}.$

Also note that OPE-key R_{ij} is shared between Voter_{*i*} and Authority_{*j*} except for R_{i0} .

Voting Phase

Each participating Voter_{*i*} ($i = 1, \dots, m$) prepares his vote as follows:

1. Voter_{*i*} decides his vote $s \in \{0, 1\}$ and compute a commitment α_i satisfying:
 $s = S_{i1}(0, 0) + \alpha_i S_{i2}(0, 0).$
2. He computes all encrypted shares $E_{ij}(x)$ for each $j = 1, \dots, N$ as $E_{ij}(x) = S_{i1}(x, j) + \alpha_i S_{i2}(x, j) + R_{ij}(x).$
3. Then, Voter_{*i*} computes a proof

$$P_i(x) = f(x)(f(x) - 1) + xR_{i0}(x)$$

where $f(x) = S_{i1}(x, 0) + \alpha_i S_{i2}(x, 0).$

(Note that this random polynomial $R_{i0}(x)$ is only known to the Voter_{*i*}, and also note that $f(0) = s$ on the second equation.)

4. Finally, Voter_{*i*} writes $i, \alpha_i, E_{i1}(x), \dots, E_{iN}(x), P_i(x)$ on the Bulletin Board.

Verification Phase

Let $V_k(i, j, \alpha) = S_{i1}(v_k, j) + \alpha S_{i2}(v_k, j)$ be a verification function for Verifier_{*k*}. Everyone (say Verifier_{*k*}) accept (or reject) the Voter_{*i*}'s vote if the following conditions satisfied:

$$\begin{cases} E_{ij}(v_k) = V_k(i, j, \alpha_i) + R_{ij}(v_k) \text{ for all } j = 1, \dots, N \\ P_i(0) = 0 \\ P_i(v_k) = V_k(i, 0, \alpha_i)(V_k(i, 0, \alpha_i) - 1) + R_{i0}(v_k) \end{cases}$$

Table 1. Storage Complexity

	Estimates	$M = 10^6, T/M=1\%$ $N = (t + 1) = 10$ $q \approx 2^{80}$	$M = 10^6, T/M=10\%$ $N = (t + 1) = 10$ $q \approx 2^{80}$
Voter	$O(NT \log q)$	3MB	32MB
Authority	$O(T^2 \log q)$	2.3GB	200GB
Public Verifier	$O(MN \log q)$	310MB	310MB

Table 2. Communication Complexity

	Estimates	$M = 10^6, m/M=10\%,$ $T/M=1\%, q \approx 2^{80}$ $N = (t + 1) = 10$	$M = 10^6, m/M=10\%,$ $T/M=0.1\%, q \approx 2^{80}$ $N = (t + 1) = 3$
1 Vote	$O(NT \log q)$	220KB	80KB
Verify&Tally	$O(mNT \log q)$	220GB	8GB

Tallying Phase

Let $U_j \subseteq \{1, \dots, M\}$ ($j = 1, \dots, N$) be the set of indices which Authority _{j} accepted during Verification Phase as a public verifier. Then, Authority _{j} ($j = 1, \dots, N$) sum up and decrypts all votes in U_j , and write U_j and $S_j(x)$ on Bulletin Board:

$$S_j(x) = \sum_{i \in U_j} E_{ij}(x) - \sum_{i \in U_j} R_{ij}(x).$$

Verifier _{k} checks at least t of U_j 's are equal. If so, let U be the agreed set of correct votes. Then, Verifier _{k} accepts the output of Authority _{j} if the following equation holds:

$$S_j(v_k) = \sum_{i \in U} V_k(i, j, \alpha_i),$$

where $V_k(i, j, \alpha)$ is a verification function for Verifier _{k} such that $V_k(i, j, \alpha) = S_{i1}(v_k, j) + \alpha S_{i2}(v_k, j)$.

Let $\mathcal{A}_k \subseteq \{1, \dots, N\}$ be the set of indices of authorities which Verifier _{k} accepted in the previous step. Verifier _{k} outputs the election result by reconstructing from the set of shares $\{S_j(0) \mid j \in \mathcal{A}_k\}$ if $|\mathcal{A}_k| > t$, otherwise outputs \perp .

4.4 Security

Theorem 4. *The above protocol is ϵ -secure electronic voting. Especially, given ϵ' -secure oblivious polynomial evaluation and ϵ' -secure publicly verifiable secret sharing, there exists ϵ -secure electronic voting, where $\epsilon < \epsilon'$.*

Proof. We have to prove Eligibility, Privacy and Integrity of the above protocol.

Eligibility is obvious, since the protocol is based on the bulletin board model where each voter is not anonymous. To prove the eligibility, it is enough to show the existence of a function f . Putting f as $f(E_{ij}) = i$ regardless of j , it satisfies the eligibility property.

Next, we prove the privacy property is satisfied by the protocol.

From the definition of privacy, the adversary is allowed to corrupt with T public verifiers and to corrupt with t public verifiers. Without loss of generality, we assume the adversary is corrupting a set of public verifiers

$$\mathbf{Y}_T = \{\text{Verifier}_1, \dots, \text{Verifier}_T\}$$

and a set of tallying authorities

$$\mathbf{Z}_t = \{\text{Authority}_1, \dots, \text{Authority}_t\},$$

respectively.

Let Voter_i be a target voter, again without loss of generality. The information posted by Voter_i is in the following form $(E_{i1}(x), E_{i2}(x), \dots, E_{iN}(x), \alpha_i, P_i(x))$ where

$$E_{ij} = S_{i1}(x, j) + \alpha S_{i2}(x, j) + R_{ij}(x)$$

for $j = 1, \dots, N$ and

$$P_i(x) = (S_{i1}(x, 0) + \alpha S_{i2}(x, 0)) \\ \times ((S_{i1}(x, 0) + \alpha S_{i2}(x, 0) - 1) + xR_{i0}(x).$$

The casted information except for $P_i(x)$ is exactly the same as that in the US-PVSS. Thus, we focus on the information leak from $P_i(x)$.

From \mathbf{Y}_T , the adversary already knows

$$\{R_{i0}(v_1), \dots, R_{i0}(v_T)\}, \\ \{S_{i1}(v_1, y), \dots, S_{i1}(v_T, y)\}, \text{ and} \\ \{S_{i2}(v_1, y), \dots, S_{i2}(v_T, y)\}.$$

The adversary cannot recover $R_{i0}(x)$ from $\{R_{i0}(v_1), \dots, R_{i0}(v_T)\}$, since $R_{i0}(x)$ is degree $2T$. Further, the adversary still has the same entropy on a target univariate polynomial $S_{i1}(x, 0) + \alpha S_{i2}(x, 0)$.

The rest of information available to the adversary is exactly the same as that in the US-PVSS. The construction described above is based on the US-PVSS construction in Section 3.2, thus it is perfectly-private from Theorem 3.

To prove integrity, we have to show that (1) the consistency of encrypted shares, (2) security of the proof $P_i(x)$ which convinces the verifiers that the secret of each voter's PVSS lies in 0 and 1, and (3) validity of the output of each tallying authority.

(1) is straight-forward from the property of US-PVSS. In the construction, we have L public verifiers. Thus, the probability that the malicious voter to put inconsistent ballot E'_{ij} to be accepted at least one of the honest public verifiers is upper-bounded by L/q from Theorem 3.

Next, we prove (2). It is easy to see that if the voter is honest, every honest verifier will accept the proof with probability 1. We will further consider the case that a malicious voter is trying to cheat at least one verifier, Verfier_k , without

loss of generality, but the malicious voter is unable to identify who is the cheated verifier. In this case, the malicious voter is trying to post a modified polynomial $P'(x)$ where $P'(x) \neq P(x)$. The verifier is trying to check the validity of the proof by checking the equation

$$\begin{cases} P'(0) = 0 \\ P'(v_k) = p_k(p_k - 1) + v_k R_{i0}(v_k) \end{cases}$$

where $p_k = S_{i1}(v_k, y)|_{y=0} + \alpha_i S_{i2}(v_k, y)|_{y=0}$. Note that $P'(v_k)$ must agree with the value computed by Verifier_k from p_k and $R_{i0}(v_k)$. Here, $P'(0) = 0$ must be satisfied. Otherwise, every verifier reject the vote of the malicious voter. Thus, we are interested in the case that second equation eventually holds for some verifier k . Assuming the v_k and the polynomials $S_{i1}(x)$, $S_{i2}(x)$, $R_{i0}(x)$ are uniformly distributed, the probability that this case happen is $1 - (\frac{q-1}{q})^L \leq L/q$.

Now we prove (3). In the Tallying phase, tallying authority Authority_j posts $(U_j, S_j(x))$ on the bulletin board, where U_j is a set of indices of the votes which the Authority_j accepted and $S_j(x)$ is a verifiable share which is a sum of every share of the votes posted for Authority_j . If all the talliers are honest, then all U_j 's for $j = 1, \dots, N$ agree with the same set unless they are cheated by the voters with negligible probability (This is from unanimity property of US-PVSS). Furthermore, every honest verifier accepts the verifiable shares $\{S_j(x)\}$ and can compute the final tally. We will consider the case that there exists at least one verifier, for example k , is cheated by some malicious tallying authorities (colluding up to t), but the malicious tallying authorities have no idea on who is cheated. Thus, the goal of the malicious tallying authorities is to cheat some verifier k with a wrong pair $(U'_j, S'_j(x))$ where $(U'_j, S'_j(x)) \neq (U_j, S_j(x))$ to be accepted.

Here, $S_j(x)$ can be written as follows:

$$\begin{aligned} S_j(x) &= \sum_{i \in U_j} E_{ij}(x) - \sum_{i \in U_j} R_{ij}(x) \\ &= \sum_{i \in U_j} S_{i1}(x, j) + \alpha_i S_{i2}(x, j). \end{aligned}$$

Verifier_k checks the validity of $S_j(x)$ by the following equation:

$$S_j(v_k) = \sum_{i \in U_j} S_{i1}(v_k, j) + \alpha_i S_{i2}(v_k, j).$$

In a case that $U_j = U'_j$, this must be the case that the malicious authority, Authority_j output a tallying result $S_j(x)' \neq S_j(x)$ and there exists at least one k such that Verifier_k accepted $S_j(x)'$. From the similar discussion as above, the probability that at least one public verifier, Verifier_k , accept the wrong polynomial $S_j(x)'$ is $1 - (\frac{q-1}{q})^L \leq L/q$.

Otherwise, $U_j \neq U'_j$. This must be the case that there exists at least one $E_{ij}(x)$ and at least one

This case happen is bounded by L/q . Thus, the success probability for the malicious tallying authorities is again exponentially small.

4.5 Efficiency

We will discuss the efficiency of the electronic voting scheme presented above. The efficiency of our scheme can be investigated in two ways: (1) Storage Complexity, (2) Communication Complexity. Since our scheme is computationally efficient. Thus we omit the evaluation of computational complexity here.

Storage complexity of our scheme is evaluated by the size of private keys required for each voter, each tallying authorities and each public verifier respectively. As described in the construction of the scheme, the storage complexity of each player is easily computed and shown in Fig. 11 under the indicated setting of parameters. The required storage size for the Voter is the most critical part. It requires 3MB and 32MB in the case of a million eligible voters and collusion of up to 10,000 and 100,000 players are allowed respectively. If some of the users want to verify the integrity of whole tallying process as a public verifier, they must store additional private verification keys listed in the lowest row. Each public verifier needs (users who requested to become a public verifier) to store 310MB of private verification keys for each case.

Communication complexity of our scheme is shown in Fig. 12 in the case of 1 million eligible voters allowing collusion of up to 10,000 and 1,000 users. Casting one vote requires only 220KB of data to post to the bulletin board. The heaviest part is the communication for the Verify and Tally to download the commitment posted by each voter (one million voters) to verify the whole tallying process.

References

1. Bennett, C.H., Brassard, G.: Quantum cryptography: Public-key distribution and coin tossing. In: Proceedings of IEEE International Conference on Computers, Systems and Signal Processing, Bangalore, India, December 1984, pp. 175–179 (1984)
2. Benaloh, J.C., Yung, M.: Distributing the Power of a Government to Enhance the Privacy of Voters (Extended Abstract). In: PODC 1986, pp. 52–62 (1986)
3. Broadbent, A., Tapp, A.: Information-theoretic security without an honest majority. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 410–426. Springer, Heidelberg (2007)
4. Chaum, D.: Secret-ballot Receipts True Voter-Verifiable Elections. In: WOTE 2006 (2006) (invited talk)
5. Chaum, D., van de Graaf, J., Ryan, P.Y.A., Vora, P.L.: Secret Ballot Elections with Unconditional Integrity., Cryptology ePrint Archive, Report 2007/270 (2007)
6. Chang, Y.-C., Lu, C.-J.: Oblivious polynomial evaluation and oblivious neural learning. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 369–384. Springer, Heidelberg (2001)
7. Chor, B., Goldwasser, S., Micali, S., Awerbuch, B.: Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults (Extended Abstract). In: FOCS 1985, pp. 383–395 (1985)
8. Cohen, J.D., Fischer, M.J.: A Robust and Verifiable Cryptographically Secure Election Scheme (Extended Abstract). In: FOCS 1985, pp. 372–382 (1985)

9. Crépeau, C.: Efficient cryptographic protocols based on noisy channels. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 306–317. Springer, Heidelberg (1997)
10. Cramer, R., Franklin, M.K., Schoenmakers, B., Yung, M.: Multi-authority secret-ballot elections with linear work. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 72–83. Springer, Heidelberg (1996)
11. Feldman, P.: A Practical Scheme for Non-interactive Verifiable Secret Sharing. In: FOCS 1987, pp. 427–437 (1987)
12. Fujisaki, E., Okamoto, T.: A practical and provably secure scheme for publicly verifiable secret sharing and its applications. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 32–46. Springer, Heidelberg (1998)
13. Hanaoka, G., Imai, H., Müller-Quade, J., Nascimento, A.C.A., Otsuka, A., Winter, A.J.: Information theoretically secure oblivious polynomial evaluation: Model, bounds, and constructions. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 62–73. Springer, Heidelberg (2004)
14. Maurer, U.M.: Conditionally-perfect secrecy and a provably-secure randomized cipher. *Journal of Cryptology* 5(1), 53–66 (1992)
15. Moran, T., Naor, M.: Receipt-free universally-verifiable voting with everlasting privacy. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 373–392. Springer, Heidelberg (2006)
16. Naor, M., Pinkas, B.: Oblivious Transfer and Polynomial Evaluation. In: STOC 1999, pp. 245–254 (1999)
17. Nascimento, A.C.A., Müller-Quade, J., Otsuka, A., Hanaoka, G., Imai, H.: Unconditionally non-interactive verifiable secret sharing secure against faulty majorities in the commodity based model. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 355–368. Springer, Heidelberg (2004)
18. Rivest, R.: Unconditionally secure commitment and oblivious transfer schemes using private channels and a trusted initializer (manuscript)
19. Schoenmakers, B.: A simple publicly verifiable secret sharing scheme and its application to electronic voting. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 148–164. Springer, Heidelberg (1999)
20. Shor, P.W.: Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.* 26(5), 1484–1509 (1997)
21. Shamir, A., Tromer, E.: Factoring large numbers with the TWIRL device. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 1–26. Springer, Heidelberg (2003)
22. Stadler, M.A.: Publicly verifiable secret sharing. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 190–199. Springer, Heidelberg (1996)

Electronic Elections: A Balancing Act

Pedro A.D. Rezende*

Department of Computer Science
University of Brasilia, Brazil
prezende@unb.br

Abstract. This article aims to share some major lessons learned from the pioneering experience in Brazil with the world's first full national implementation of universal electronic voting. Differing notions of security, and their "collateral entanglements", appear to play a key role and are contrasted in Brazil's pioneering electronic voting saga. After an introduction, we puzzle through what election security may mean. We elaborate on how technological innovations may affect the underlying risks, their nature, corrections and balance. Then we describe some ways in which innovations have been deployed and validated, and how the results are being perceived, before some closing remarks.

Keywords: Electronic Elections, Electronic Voting, Voting Auditability.

1 Introduction

Four times since 2000, until the writing of this article, more than one hundred million voters in Brazil have been obliged¹ to vote using direct-recording electronic voting machines (DREs), which do not allow for recounts. This raises questions such as whether, and if so how, electronic elections can be audited meaningfully. Such questions have been the subject of academic debate worldwide, and in Brazil the discussions started even before DREs were fully deployed².

The real issue is auditability. That is to say, the nature of possible assurances regarding the correct tallying of the votes cast by entitled voters. This boils down to the pertinence, or necessity, for a material representation of each vote to be held by the voting system, to allow for credible audits. Credible, here, meaning worthy of trust by

* Pedro Antonio Dourado de Rezende is a tenured professor at Computer Science Department, University of Brasilia (UnB). Advanced to Candidacy for PhD in Applied Mathematics from University of California at Berkeley in 1983, heads the UnB Cryptography and Info Security Extension Program since 1997. Author of over one hundred articles on related topics, was a member of Brazil's Public Key Infrastructure Steering Committee from 2003 to 2006, as representative of civil society by presidential appointment.

¹ Voting in official elections in Brazil is mandatory for eligible voters of ages 18 to 65, under national electoral law.

² In Brazil, electronic voting machines were introduced in 1996. However, debates on electronic voting audit had already started in 1982, with the first reported case of electronic fraud in vote tallying. This happened in the gubernatorial election at the state of Rio de Janeiro, also the first to be tallied electronically, in what became known as the ProConsult case. [see ref. 1].

the technical non-savvy. In the U.S., where local political subdivisions have significant autonomy in how they run elections, the debate started in the mid 70's, picked up some visibility in the 80's, and gained global headlines with the 2000 Florida results. In Brazil, where federal law defines election processes uniformly, the debate gained equivalent attention twice, though each time only briefly, in 1982 and 2001³.

Many computer security experts from the U.S. and Europe participate in the U.S. debate. In Brazil, in spite of the pioneering and uniquely universal use of DREs, involvement of experts in the debate has been quite limited. In either case, however, those in charge of running elections also have a point to make, mostly divergent from the experts'.

2 The Puzzle of Election Security

Officials responsible for organizing and running elections have been, for instance, largely against audit measures based on voter-viewable printouts. Some have been quite vocal about it, as in Brazil, presumably because of the inconvenience such measures might impose on their work⁴. But surely also because, although few would publicly admit it, eventual discrepancies between electronic and equivalent manual tallies would allow discovery of casual ineptitude, or even possible bad faith, in the discharge of their official duties. On the other hand, such audit capability would also diminish whatever bully power, explicit or implicit, such officials might wield (or intermediate) among elected politicians and aspiring candidates or their political parties.

However, most independent information technology experts who have written on the subject⁵ have tended to favor the requirement that each electronic voting machine be set to print a record of each vote, with the printed record visually checkable by the voter. The reasons for this opinion, explored more fully below, include anchoring

³ In 1982, with the ProConsult case (previous footnote), and in 2001, with the "Senate's panel scandal", briefly covered ahead. [for a thorough account of the latter, see ref. 2].

⁴ Several electoral officials in Brazil, including judges, have publicly opined that this kind of audit measure constitutes "retrocession." [see ref. 2].

⁵ Aviel Rubin, <http://avi-rubin.blogspot.com/>; Bruce Schneier, <http://www.schneier.com/crypto-gram.html>; Douglas Jones, <http://www.cs.uiowa.edu/~jones/voting/cbc2004supp.shtml>; Dan Wallach, <http://avirubin.com/vote/analysis/index.html>; David Chaum, http://people.csail.mit.edu/rivest/voting/papers/CryptoBytes_Fall2004.pdf; David Dill, <http://securingamerica.com/ccn/node/8023g>; Ed Felten, <http://itpolicy.princeton.edu/voting/audit07full.pdf>; Michael Waldman (Editor of the "Brennan Report"), http://www.brennancenter.org/presscenter/releases_2006/pressrelease_2006_0627.html; Rebecca Mercuri, <http://www.notablessoftware.com/evote.html>; Ron Rivest, <http://people.csail.mit.edu/rivest/Rivest-TheThreeBallotVotingSystem.pdf>; Roy Saltman, http://www.vote-fraud.org/saltman_roy_1988_report.htm; Robert Strunk, http://www.vote-fraud.org/expert_strunk_report.htm

convictions of electoral results' correctness in the participation and experience of individual common voters, as something essential for the voter confidence which – we believe – underlies the spirit of democracy.

From the technical standpoint, these experts may defend the retention of some material representation of individual votes by electronic systems for another simple reason: if they are convinced that the scientific resources and technological tools available to, or even possible for, computer security are insufficient to sustain trust in the outcome of fully electronic secret ballots, at least to an extent consistent with the spirit of democracy.

Among these experts we find living icons of Computer Science, such as Ronald Rivest (one of the inventors of the pioneer RSA method for digital signature), David Chaum (inventor of eCash “digital cash”) and Bruce Schneier (cryptographer and author of major best-sellers on computer security).

Their reputation has led, for instance, at least one political scientist to argue why it is much easier to protect financial electronic transactions against electronic fraud than to tally a fully electronic ballot of secret votes with equivalent overall security [17].

Reliance on fully electronic mechanisms for voting and for election auditing purposes yields more routes for plausible deniability to those who may wish to stealthily interfere in the electoral result while controlling the underlying technology. Relying solely on electronic measures for auditability has meant that any new measure designed to close these routes end up opening their own.

As a contribution to this debate, we posit that the heart of the disconnect between these two groups – formed by distinguished computer security experts and by election officials or suppliers in favor of fully electronic voting systems – may stem from the different way that each group, either by virtue of their craft or by gut feeling, understands “security”:

- **[1st sense]:** security from the standpoint of voters (and experts on their behalf)
 - a) with rights to a secret ballot and to its correct tallying,
 - b) against possible manipulations of the electoral process,
 - c) by whoever in the electoral system,
 - d) which should be readily detectable by voter oversight;
- **[2nd sense]:** security from the standpoint of those running elections
 - a) with rights to program or operate the electoral system,
 - b) against detection by voter oversight,
 - c) of whatever act imputable to ineptitude or bad faith,
 - d) through which manipulations of the tallying is possible.

3 Risk and Modernity

The main difficulty we can point to, regarding the security of fully electronic voting systems as we see it, is rooted in an inconsistency between two basic requirements. The first of these requirements is vote secrecy, and the second is the requirement for dematerialization of votes (if the voting system is to be fully electronic). The inconsistency, explained in the next section, arises under real-world conditions, in

the context of real democracies, from the fact that at least three potentially conflicting interests are at stake in elections: the interest of voters who believe in, or desire, democracy through fair and clean elections, and the interests of at least two competing candidacies.

To understand how real life conditions make those two requirements inconsistent, one needs to note that election integrity can only be guaranteed if voters are also protected against manipulations of internal origin, which is to say, if operatives of the electoral process who may stealthily favor such tampering are to be, for that purpose, unprotected. This means that the first sense of security cited above, a legitimate sense from a perspective we believe to align with the spirit of democracy, can only be effective if coupled with the suppression of the second, an illegitimate sense from this perspective.

On the other hand, attempts to have a voting system fulfill both requirements (vote secrecy and electronic dematerialization) at once, while formally aiming to achieve that first sense of security, may – or will, as we'll argue – yield the practical effect of reaching out for the second. This would turn the risk profile of such systems unstable. Thus, one may begin to understand how technical discussions which bypass the need to extricate these two senses of security will likely degenerate.

A debate that fails to extricate these two senses of security will cloud the possible tracks through system design choices along which risk estimates can be reasonably expected to remain constant. This problem is aggravated when, from a position of authority further empowered by a choice for vote dematerialization, electoral officials in favor of fully electronic systems willfully ignore this analytical imperative, at the guise of specious, faulty or bogus arguments, mostly non-technical.

This analytical imperative stems from the fact that processes with more than two potentially conflicting interests at play (as with any electoral process) pose risks of a kind known as collusion. These risks have in common the fact that they vary when security is sensed from the perspective of different interests. A typical collusion requires two or more parties, engaged in the process, to disingenuously act as if their interests diverge, in order to reach a disguised benefit to some interest they stealthily share, at the expense of some illegitimate harm to a third interest.

In electoral processes, collusion can happen through secret alliances, in which uncompromising conflicts of interests (or independence of actions) are faked. Or they can happen the other way around. In short, electoral processes exist under the systemic, intrinsic risk of collusions, either by fake conflict or by fake cohesion of interests or actions, aimed at harming other interests in order to improve the colluders' chances for later sharing power, more power or its bounty.

Therefore, to blur paths where risks can conflate as they spread, making them appear as diluted, is not conducive to good policy or sound analysis, as the current global economic crisis, stemmed from financial innovation in free markets, is now showing.

4 Balancing Risks

Vote dematerialization enrich the ways through which risks of collusion can compound and materialize, by offering colluders new means to hide their methods, if

enough electoral authority is inept or involved. Labeling such considerations as “paranoia” or “conspiracy theory” will not make these facts go away; rather, such *ad-hominem* rhetoric signals that the extrication of those two senses of security is prerequisite for a thorough, balanced analysis of the e-voting modernization phenomena. But the persistence of such *ad-hominem* rhetoric, specially by the mainstream media in chorus with the discourse of electoral officials and suppliers, also yields a constructive reading. It reminds us that collusion strategies can, of course, start with obfuscation of main motives for certain choices in the design and procurement of voting systems, in tandem with lobby for electoral regulation reform to legitimize them. If so, such strategies need to drive those two senses of security to appear indistinguishable, or inseparable. From there, to a collusion's full feast is an easy ride: through the disguising of the second sense of security as the first, say, as an inevitable consequence of technological progress.

Thus, the security of legitimate interests in representative democracies, at least in democracies bound to preserve the spirit of its humanist revival⁶, ought to be sought by fully acknowledging and considering not only risks of collusion, but also the ensuing profile of risks and how this profile can change with changes in the electoral processes and in voter mentality. And not to be sought by unilateral control of the process, be it by the market's invisible hand or any other, or through jealously guarded secrets of its mechanisms, which over-empower the beholders.

For its part, adequate protection against collusions can only be achieved with adequate balance between transparency of the (electoral) subprocesses and distribution of their controls among legitimate and potentially conflicting interests, integrated in a way to allow for an effective oversight. In electoral processes, or in any other process intrinsically exposed to the risk of collusions, the more technological intermediation there is the more such balance will hinge on two basic elements: Carefully tailored regulation, and participation of stakeholders (voters) in the oversight process.

This is of the utmost importance for elections, due to two main thrusts. First, the risk of collusions as a constant menace to representative democracy, due to its delicate political nature. Second, technological intermediation as a wedge, parting voters from autonomous oversight roles and, ultimately, risking their role as democracy's guarantors of last resort. In our view, backed by the empirical evidence given here, inconsistent voting system requirements can lock these two risks in positive feedback. And in our times, there seems to be nothing more effective for this than the requirements of vote secrecy and of vote dematerialization through complete computerization.

For historical evidence, on the first thrust we cite the comprehensive research by John Fund published in 2004 [15], regarding the U.S., the nation with yet the most successful case of democratic rule⁷. And on the second, plus empirical evidence on how these two thrusts may feedback, we offer the last sessions, regarding Brazil, a hesitant latecomer to democratic rule. On feedback signs we pick some from a collective spell of supposed technological prowess Brazil seems to

⁶ From the French and the American Revolutions of the Eighteen Century.

⁷ At least in the sense of being the nation with the longest continuous period of democratic rule.

be under, given its modern voting system, where most voters seem oblivious to the lessons from their Old Republic⁸.

5 Collateral Entanglements

In secret ballots, that is to say, in ballots requiring a voter's identity not to be associated with his or her vote during casting or tallying, the electoral supervision process becomes, due to this vote secrecy requirement, sensitive to the physical way in which each vote is cast. As a consequence, if the electoral process dematerializes the votes, recording only by digital means partial tallies of votes cast, whatever oversight process the system may feature seems to end up ineffective, as if "tied up".

Tied up in the sense that any oversight measure aimed at detecting or deterring insider malfeasance (that is, malicious acts by electoral operatives in possible collusion with some candidate) will also serve to protect outside defrauders, that is, voters overseeing the process for a candidacy willing to sabotage the oversight process (to call maliciously into question an election deemed lost) or to subvert it (to insert defrauding mechanisms into the system).

Whereas, symmetrically, any measure to detect or deter sabotage or subversion in the oversight process will also serve to protect malfeasance by insiders holding privileges to program or operate the system. These entanglements between intended and collateral effects, observable (as reported below) from Brazil's experience with its fully electronic voting system, raises the central question for this article: Is this observed pattern of "collateral entanglements" due to inept implementations of security measures in a particular fully electronic voting system, or due to conflicting voting system requirements?

Computer scientists who want to seriously study the computerization of elections shall not allow for ideologies to obfuscate the contours of the problems under focus, inherent to voting systems, but rather, they shall distinguish ideologies as a source for them, in so far as ideologies shape the social and political value of elections. From this perspective, the scientific study of electronic voting systems reached a milestone in 2000, with a PhD thesis successfully defended at the University of Pennsylvania [16]. In her thesis, Rebecca Mercuri is believed to have demonstrated

⁸ In Brazil, where the widespread use of DREs was pioneered from 1996 on, history books – and Wikipedia – explain how the nation's first period of democratic rule, from 1989 to 1930, known as "the Old Republic", was plagued by collusion. Election organizers and two main political groups, led by landed gentries, were involved. Regardless of the real outcome, the books were cooked at each election so that the two groups would alternate at filling the country's presidency, while the three pretended, with help from the fourth power – mainstream media –, to find no wrong through the electoral supervision process. The Old Republic's plot became known as "política café-com-leite" ('cappuccino' politics), from which voters took decades to realize detrimental consequences. This, in turn, led to civil unrest and a coup, the 1930 revolution, to reform democratic rule. After two interruptions of democratic rule, (from 1937 to 1945 and from 1964 to 1988), now under the spell of some supposed technological prowess, most Brazilians seem oblivious to the lessons from their Old Republic. Like their neighbors from Paraguay, where Brazil's DREs has been borrowed, but unlike their neighbors from Venezuela, whose later debut with democratic rule was plagued by similar plot, from 1958 to 1998, known as "pacto del Punto Fijo".

that vote secrecy and tallying integrity are mutually exclusive guarantees that a fully electronic voting system can offer.

In other words, with fully electronic elections, there is no way to have vote secrecy and tallying integrity protected in the same run because, to use a simplifying metaphor, these promises are like two sides of a coin. A coin representing the electronic voting system, with value corresponding to that of the electoral process it can execute, but a coin that cannot be "flipped" to show both sides during an election because it executes the election in a single run, without the possibility of recount for auditing purposes (due to dematerialization of the votes).

One can argue about whether and how Dr. Mercuri's work can rigorously lead to such conclusion⁹, but the weight of its scientific arguments can be felt in many fronts. For instance, as an answer to the central question raised here, from Brazil's pattern of "collateral entanglements". Or, in electoral legislation across the U.S., under pressure from civil and grassroots movements, specially after dubious ethics from main DRE suppliers began to surface [3], [4]. Between March 2004 and May 2005, fourteen federated states approved laws requiring voting machines to allow for Voter-Verifiable Printed Audit Trails (VVPAT), to retain or recover the supervising capacity common voters unquestionably had before elections were computerized.

Before July 2006, 27 U.S. states have such laws already sanctioned, thirteen of them with mandatory manual audit. Only fifteen U.S. states appeared to see no problem yet with DREs. As for the U.S. Congress, several bills aimed at assuring that VVPAT becomes a federal tenet for electoral supervising processes are being considered. This, not to naively pretend to do away with election fraud, but to put all of their forms in a hard-playing leveled field, that is, to expose the ways to defraud – old and new – to the risk of detection by common voters in due time. In other words, to give back to common voters – with no PhD in Computer Science – their legitimate right to supervise elections with autonomy.

6 Routes to Electronic Elections

Each democracy has to answer the call to go modern, if for no other reason then because of the massive lobbying by DRE suppliers, the larger of which has gone global. A special look at the route taken by Brazil seems warranted, if not because of its pioneering widespread use of DREs, then because of the keen interest its system has raised within the Organization of American States (OAS), or because Brazil's deployment took a route leading to a landscape quite distinct from what has been portrayed by lobbies, by Brazil's mainstream media and by specialized global media, and perhaps also quite distinct from the route the U.S. seems to be taking¹⁰.

⁹ In the same year Mercuri defended her PhD thesis, for instance, Berry Schoenmaker published the article "Fully Auditible Electronic Secret-Ballot Elections":

<http://www.xootic.nl/magazine/jul-2000/schoenmakers.pdf>

¹⁰ As far as we can tell, Brazil is unique among modern democratic republics in concentrating, in a single institution, the electoral functions from the three powers a republic should keep separate, namely, those to legislate, to execute and to adjudicate. This institution, called 'Electoral Justice', is organized as a branch of the Judiciary, and binded only by the Constitution and federal election laws. The Constitution and electoral laws compel all statutory, executive and adjudicative matters regarding official elections into a Kafkean system of 'electoral tribunals', one for each state, all under a federal 'Superior Electoral Tribunal' [TSE].

In Brazil, the highest electoral authority – the *Tribunal Superior Eleitoral* (TSE) – has picked one model of voting machines to serve all 400 thousand plus precincts in the country, has procured, deployed and put to use such machines nationwide since the municipal elections held in 2000. TSE has designed its voting system around the voting machine model it has picked, which is a type of DRE with one added twist: a terminal used by precinct officials to check voter identity physically connected, by a 12 ft. cable, to the voting machine itself.

The voter ID number is typed in this terminal, to be checked by a software running on the voting machine. This ID is checked against a list of registered voters allowed to vote in that precinct, kept in a file stored alongside the file with the vote tallies, in a voting machine's storage media. If an entry is found with that ID, and if the entry isn't marked with “already voted”, the software shows the voter's name on the terminal's single-line display and the machine is allowed to receive a vote. Otherwise, an error message is displayed. Thus, given the current oversight rules and practices, this choice of design makes vote secrecy an act of faith in software (non-)functionality.

From 2001 on, the political input into Brazil's voting system's design began to change. In May of that year, from a collusion among top senators gone sour a case of electronic voting fraud in Brazil's Senate¹¹ broke out in mainstream media, causing a great deal of public outrage. Besides how easy it was for operators to violate the secrecy of votes, the scandal also unveiled how fully electronic voting systems can be resourceful for colluders. Public indignation then pushed the Congress to take up the matter of revising election law, so that recount mechanisms would be introduced for general elections.

A Bill to that effect was introduced¹², but encountered fierce resistance from the authorities whose activities would be monitored under its provisions. The Bill's passage was targeted for disruption by the president of the TSE, in a series of actions that drew no attention from mainstream media. First he asked the Senate, in his capacity as the head of the highest electoral authority (appointed by, and from among Supreme Court Justices), to await for input from his institution. To deliver, he waited until five days before the constitutional deadline for passing the Bill if it were to have effect during the next elections, reminding senators of this urgency (meaning, no floor debate).

Among the proposed amendments he sent to the Senate, on plain paper with no official letterhead, one effectively did away with the printed vote function, by providing for prior selection, on election eve, of the voting machines to be used as sample in mandatory recount for audit purposes. The senator who sponsored these amendments and lobbied his peers for approval, under loose rules for matters declared urgent, was awarded by TSE, two weeks after the Bill was approved with this crippling amendment, a 15-month mandate as governor of his state¹³.

¹¹ A case of legislative vote fraud known as the “Senate panel scandal”. [see ref 2].

¹² Senators Roberto Requião and Romeu Tuma introduced a Bill mandating that the DREs be adapted to run VVPAT extension modules, for unencumbered tally audit by manual recount of a 3% sample of precincts.

¹³ A court case that had dragged on in TSE for more than two years, over a bid in which senator Hugo Napoleão had ran for governor of the state of Piauí in 1998. The declared winner had been governing for more than half the mandate, but the election was impugned, based on a claim that the winner's campaign finances were not up to snuff.

Then, after that deadline had elapsed and the crippled Bill was in the lower house, the same president of the TSE – where electoral laws are interpreted – changed the story, suggesting that better it be voted on as an urgent matter¹⁴, arguing that the matter could still go into effect for next election since it was, “after all, a technical matter”, and therefore beyond the constitutional restriction for electoral matters, of prior approval by one year.

After the crippled Bill was passed and sanctioned as suggested, becoming Law number 10,408/02 (VVPAT Law), he invited some Congressmen to his office at the Supreme Court to inform them that he had misunderstood the constitutional restriction: such legal matter was indeed electoral in nature, and therefore the VVPAT Law would not apply to the next election. As an excuse for his fumble, he offered to have electoral authorities voluntarily “test”, in 3% of the voting machines at the upcoming 2002 election (which included a bid for Brazil’s presidency), the VVPAT mechanism that such Law had made, as he understood it then, obligatory only for elections scheduled to be held after 2003.

For this “test” he would order the adaptation of only some of the existing DRE machines, expanded to allow the appendage of a VVPAT device (image below), as proposed in VVPAT Law’s justification.

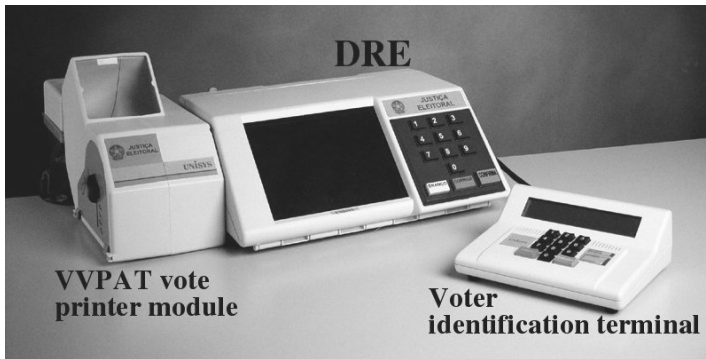


Fig. 1. Brazil’s 2002 DRE with VVPAT module from www.unisys.com.br/img

7 Political Design Validation

The guest legislators accepted the offer, allowing the target of supervision to “test” a mechanism which Congress had chosen for monitoring their activities, and “test results” could be observed. Due to the purpose of this work, we’d rather mention what the mainstream media didn’t: Failures in the instructions for how to set up the (vote) printers, failures in voter training (voters needed to press “confirm” one more time, but weren’t told that), failures in voter registration (careless excess of voters registered precisely to precincts that featured printed vote without proper instructions) [10].

¹⁴ Again with no floor debate, and with no further amendments so that the Bill wouldn’t have to go back to the Senate.

Failures that led to long lines, frustrations and problems, failures ignored both by mainstream media and by a self-evaluation that the TSE later published about such "test." Problems that the TSE self-evaluation and mainstream media blamed, as if obvious, on the audit measure itself, not on the conflict of interest in having electoral authorities test a mechanism that legislators had chosen for voters to supervise their power. This self-evaluation was prepared and presented to Congress, in 2003, by the TSE president who not only ran this plot, but also, as a congressman in the constitutional assembly of 1988, admittedly smuggled articles into Brazil's Constitution [13].

Based on this TSE self-evaluation, a senator with unclean record¹⁵ then proposed a Bill that would amend the VVPAT Law so as to eliminate the VVPAT audit measure. The last shred of voters' right to recount votes after the computerization of elections was to be eliminated before it was ever exercised. To replace it, the senator offered Brazilian voters what he called "digital vote registry." As a justification for his offer, we learned that:

"The substitution, proposed by the current Bill, of the printed vote by the digital record of the vote for each office, with the identification of the voting machine on which the vote was recorded and the possibility of recovering it, perhaps for future analysis, while protecting the voter's privacy, will without a doubt increase the security and transparency of the elections process, making the printing of a record for the voter to check a dispensable measure."

Just like the crippled version of the VVPAT Bill, this amendment also passed with no floor debate and with no public hearing [8]. As to the "transparency of the process", not a chance: every plea made thus far by election supervisors to access the encrypted "digital vote registry" has been denied "for security reasons" [14]. Meanwhile, Brazil's on-again, off-again main supplier of DREs¹⁶ has been acquired by a company that has been selling DREs of the same basic design¹⁷ in the U.S., as code leaked from both reveals [5], [6], [7].

8 Reductionism

Several documents indicating serious security (in the first sense) flaws plaguing Brazil's voting system¹⁸ were made available to lawmakers, as they considered amending

¹⁵ Sen. Eduardo Azeredo, who has been indicted in Federal criminal court for allegedly masterminding the money-laundering and embezzlement scheme that became known as "Valerioduto" [see ref. 14].

¹⁶ Procomp, an IT company formerly owned by Brazil's largest domestic bank, later bought up by the largest U.S. supplier of DREs in late 2007, Dieblold.

¹⁷ Except for a new outfit and no VVPAT extension or dangling voter ID input modules.

¹⁸ These documents include a manifesto and petition by university professors warning lawmakers and the public of major risks inherent to fully electronic voting systems, which do not allow audits of the electoral process, asking that debates to legalize them include public hearings; a Technical Reports from the Brazilian Computing Society (SBC) and from Coppetec, a technology research center from the largest public university in Brazil, the former recommending the use of VVPAT modules in voting machines to allow for unencumbered tally audits by manual recount of a sample of the precincts; an Expert Report on a DRE from a Santo Estevão precinct, part the electoral lawsuit case TRE-BA 405/2000. This is a document produced for lawsuit in which two right-wing parties litigate over the result of Santo Estevão's 2000 municipal election.

the VVPAT Law, but the indications were dismissed. These indications were later corroborated by source code leaked to the Internet, which turned out to be part of the software used in voting machines in Brazil's 2000 municipal elections, according to an analysis done by the author [6], comparing with code later appended to an expert report filed in a court case, in a lawsuit over a disputed municipal election known as the Santo Estêvão case [7].

The code analyzed was the part which controls security for the DRE software (`setup.bat` file, in Brazil's 2000 voting machine model). The analysis revealed how ineffective the electoral oversight process was [6]. Despite the importance of such findings, they raised no interest with mainstream media or the general public. However, the Santo Estêvão expert report is extremely important because it documents the only independent technical analysis yet permitted on voting machines used in official elections in Brazil.

The report reveals, for example, how the physical seals for the DRE machine, which purportedly guarantee them against tampering after software installation, were absolutely ineffective in the first sense of security cited above, while absolutely effective in the second sense¹⁹. Four physical seals were prescribed, in pedantic details as to the positions they should be placed, by an official bylaw²⁰ which was amended as soon as the Santo Estêvão expert report was filed. This amending was done with backward dating, so that corrections appeared to have preceded the independent expert findings.

This security flaw in Brazil's electronic voting system was acknowledged by authorities only because the obscurantism surrounding the system briefly lapsed, when the Santo Estêvão's judge allowed an independent expert witness to examine voting machines. Yet, this cluster of facts does not connect dots in the public mind. Most people confuse such obscurantism with security, and this lapse of obscurantism with breach of security (as a breach by the expert witness).

The report also reveals how the language of electoral bylaws, under such obscurantism and leveled by official boastings about the security they warrant, can shed light on the main questions raised here: on the nature of "collateral entanglements" in fully electronic voting systems, on how inconsistency in system requirements can entail such entanglements, and how these entanglements can feedback risks. It reveals, in other words, how that second sense of security can be disguised to appear as the first, through a discourse of authority. This episode has, in our view, the value of a cornerstone in understanding how such deceitful collective perception is build: weaved of flag-waving vainglory, of collective ignorance and of conceited arrogance, into a pattern of reductionist beliefs.

Most victims of such reductionism so become by cutting corners in understanding what is at stake. By mixing up electoral process and electronic voting, or by confusing vote secrecy with secrecy in the process of collecting and tallying votes. Or, by naively believing in rough conjectures about what transparency means, or how much of

¹⁹ The seals, if placed as prescribed, are left intact when the DRE cabinet is open by releasing a screw hidden behind the DRE's mounted battery. This would give access to the DRE physical storage (flashcards). On the other hand, any unauthorized access to voting machines, say, to unmount the battery and inspect the DRE, is a Federal crime.

²⁰ TSE Resolution n° 20.966.

it is enough in this process. Others so become by not knowing what good transparency could do when computers take over, and others, by being clueless about how much more important it becomes in these cases. Yet others, by believing in hunches about why more transparency would hurt security, in a vague and undefined – if not Manichean – sense: the hackers!

To aggravate, there are “specialists” with thin scruples and cloudy ambitions always ready to explore such reductionism, as if voting machines were akin to magician’s black boxes [9]. Thus, the urge to breach the dogma of security through obscurantism, frequently disguised as technicism, to reveal how fully electronic voting systems entangle legitimate and illegitimate senses of security. Those two senses of security cited here are not the same, in fact each can only be effective with the suppression of the other. As to which will prevail, this is up for grabs when voters don’t care to participate in the process all the way to the level of autonomous oversight. Not easy, because dogmas are powerful.

9 Evoking the Holy Byte

Fully electronic voting systems would have marveled Machiavelli, had they been available around his time. By the exuberance of belief patterns they seem prone to elicit, towards some kind of techno-messianism. The one sprouted in Brazil has been called “the creed of Saint Byte”, a pun with a local creed (pun translated as “the holy byte”)²¹. These patterns evolve with the dogmatization of some conjectures, circulated as commonsensical truths by mainstream media. Some conjectures are about how much transparency is good for electronic systems, with voting systems as a test bed for the faith.

The creed of the holy byte purports to reveal how this leave-it-to-the-experts type of reductionism can save Brazil’s democracy from human sin. By spreading the faith in the inseparability of those two senses of security [10], [11], the faith that put designers, deployers and operators of such systems in a straight path to digital sainthood. The faith in the power of electronic purity, which shall free us from that evil plaguing civilizations for millenia: the diabolical, inefficient paper. Free at last!

Perhaps due to its pioneering in electronic vote, Brazil is coming out as a copious source for signs of this techno-messianic phenomenon. One has only to ingest the potion²² offered by local mainstream media, through eyes and ears at the electronic altar of consumerism, to reach a Mystical Vision in one’s own home: angelical beings designing, programming, configuring and operating DREs.

²¹ *Translator’s note:* Creed of Saint Byte is a parody of *Seita do Santo Daime*, pronounced alike in Portuguese. The latter is, citing wikipedia, “a syncretic spiritual practice, which grew out of the Brazilian Amazonian state of Acre in the 1930s and became a worldwide movement in the 1990s. Practitioners of Santo Daime (who call themselves Daimistas) believe strongly in the spiritual benefits of drinking the sacramental tea Ayahuasca, [which may be classified as halucinogen], in the context of rituals, or spiritual works. Santo Daime can be understood as part of the rich spiritual landscape of religion in Brazil.” To follow the pun’s intention, we translate to “holy byte”.

²² From the parody of the “Seita do Santo Daime” [see previous footnote].

Fig. 2. 1987 advertisement: "Without it, life would be hell"

To exemplify, we cite two impressive signs. One, the continuing veto by TSE of requests to allow for independent homologation procedures, prescribed by well-established technical standards for electronic information systems²³ (such as the International Standard Organization), on Brazil's official voting system. Two, the suppression of the only means by which voters could independently verify the tally, for any eventual manipulation therein, in the bylaws for the 2006 elections: ballot reports, printed and signed on paper by precinct officials at the end of voting period, shallhenceforth not be handed out to more than "one representative of the political parties"²⁴.

The alleged explanation for the first of these signs, for the shutdown of doors to independent homologation, is the self-serving argument that electoral bylaws (written by the system operators themselves) do not prescribe such tests. The only tests allowed, labeled as audit, as oversight or as independent homologation to suit the occasion, are

²³ Among signs of this revelation we can cite: a dogmatic contamination of technical studies on the security of Brazil's electronic voting system, ordered and paid for by Brazil's main electoral authority -- TSE --, such as the 2002 report "from Unicamp" in light of an independent analysis of the 2000 *setup.bat* file [see refs 6, 7 and 9]; a veto on the participation of Rebecca Mercury in a scientific meeting on electronic v, sponsored by TSE and the University of Santa Catarina in 2003, under the allegation that her views would have, according to a witness, "nothing to contribute to the betterment of our system"; a systematic refusal of TSE to allow any independent homologation of the voting system, by voters or by technical assistants to candidacies, not even as prescribed by national or international industrial or commercial standards such as ISO's for Information Systems' Security.

²⁴ As per § II of Art. 42 of Resolution n° 22.154, issued by TSE on May 2006, later amended by Resoluio n° 22.332, issued on August 8, 2006. Some state authorities, like São Paulo's (through TRE-SP Instruction n° 12.523 of Sept. 22, 2006), have directed precinct officials at the 2006 general election to ignore that late amendment, and thus, to deny printed ballot reports to representatives of single political parties.

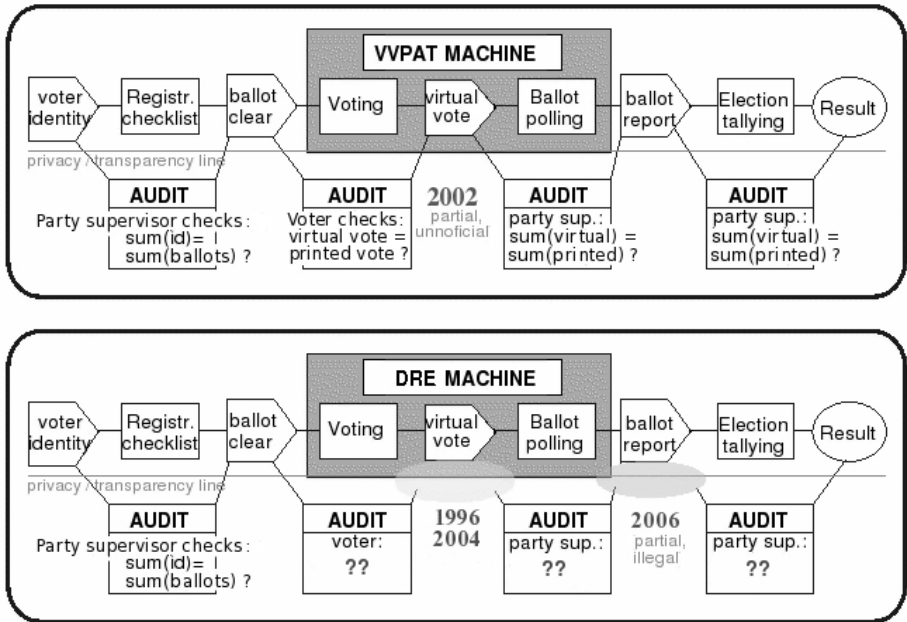


Fig. 3. The route of Brazil's voting system model

the ones their own wisdom define, which amounts to overseers' mere hands-off observation of DREs emitting reports of self-indulgence. And for the second sign, for the shutdown of doors to tallying verification by disgruntled candidates or skeptical voters, the explanation is to expedite the proclamation of election winners and to save paper.

None of these signs of techno-messianism seem to wake up the mainstream media to their investigative journalistic value, even as fables. Rather, Brazil's mainstream media has been busy with the self-appointed task of protecting the masses from the risk of "losing trust in our system". For that, it endlessly recites, preferably through the mouth of some higher electoral authority, mantras from the creed of the holy byte. Such as: "our pioneer electronic voting system is 100 % secure, for if it was not, proofs of fraud would appear before us!".

While the holy byte dogmas circulate as self-evident truths, the real debate over the security of electronic voting systems is, to the general public, skewed or muted. While the new means to defraud elections entailed by fully electronic systems, bearing stealthier and more concentrated swindle power than ever, keep getting disparagement or silence from the fourth estate [12]. While the argument of tallying agility as justification for this rationale remain bogus: France and Germany tally faster with paper ballots than Brazil does with DREs.

Moved by a creed untold as such, mainstream media now behaves and report as if elections have become (except for proportional races) some sort of video game. The voter is invited to watch a sort of poll-driven virtual race, with the checkered flag falling on election day. In the final lap, the voter goes up to a black box and pushes some buttons, then sits down in front of the TV to see the results. "Experts" take care of the rest. The importance of autonomous voter oversight to the process

has disappeared from public awareness. Nonetheless, lessons from Brazil's Old Republic²⁵ were not forgotten by all.

10 History Lessons

Those who heed History can observe – and in this case report – double standards being, again, applied to electoral matters. Given the aim of this article we now focus on the American continent, especially on a self-appointed role played lately by the OAS, the role of some sort of “democracy police”.

Of the only country to have yet adopted VVPAT as uniform requirement for its electronic voting system²⁶, OAS officials demanded, in an election held there in 2004, that the final tally be audited by manual recount in a sample they would help pick, of 1.5% of the precincts. At the end, 54% of the precincts were audited clean by manual recount. This was a referendum that could have toppled an elected president at the middle of his mandate. For the rest of Latin America, however, OAS encourages, or engages as a broker for, the use of Brazil's electronic voting system, which does not allow for recount. The same system whose designers, operators and lobbyists fight hard to never allow to become effectively auditable, to the point of even defrauding the legislative process which sets its main requirements²⁷.

To brag about this engagement, TSE has even published a booklet with a list of countries OAS is helping get used to, or get to use, Brazil's electronic voting system²⁸. Argentina, Costa Rica, Dominican Republic, Ecuador, Mexico and Paraguay²⁹. However,

²⁵ The flag from Brazil's state of Paraíba is a homage to a former candidate for Brazil's vice-presidency, João Pessoa, whose assassination is believed to have sparked the 1930 Revolution, a popular revolt that busted the enduring collusion plot known as “política café-com-leite” (“cappuccino” politics). Paraíba's capitol is also named after him. According to the state's official web site, the red part of the flag stands for the blood shed in his assassination, and the remaining black for the mourning feeling after his death. The word NEGÓ, which means “I refuse”, over the red part refers to the ensuing revolt against the “café-com-leite” collusion practice, fueled by the state's refusal to accept the official 1930 presidential election result. That result had declared the defeat of João Pessoa, governor of Paraíba at the time, and of his presidential mate, Getúlio Vargas. The rebellion set in motion by Pessoa's assassination, before inauguration, ended up conducting Vargas to the presidency, exposing a distance which legality can stray away from legitimacy.

²⁶ Venezuela: <http://www.cne.gov.ve>

²⁷ The law revoking the undebuted VVPAT measure (Law 10.740 of 2003) and the law introducing electronic voting systems into Brazil's electoral process (Law. 9.100 of 1995) were admittedly drafted by TSE staff. Under constant lobbying by electoral officials, the corresponding Bills were voted by the two houses of Congress, passed and sanctioned into Law by the president in record time (less than six months), with significant engagement of politicians involved in electoral litigation and not a single public hearing or amendment allowed. Throughout the process of drafting, discussing, voting and sanctioning them, any and all contributions offered by the academic community were ignored. Law nº 10.740 was approved by Brazil's Congress in September 29, 2003 with grave irregularities, documented in www.brunazo.eng.br/voto-e/textos/PLazeredo.htm (see [ref. 8]).

²⁸ “Informatization of Brazil's Electoral Justice” (Informatização da Justiça Eleitoral Brasileira), TSE, Brasília, 2005.

²⁹ By the time the booklet was published, presumably in 2005.

Argentinian judges have three times blocked the use of Brazil's DREs in official elections, in 2001, 2003 and 2005, allegedly because the machines did not allow for manual recounts or tally audits. In Mexico the offer was turned down, if for no other reason because some states there have been using VVPAT machines. Paraguay has been the only other country (besides Brazil) to have yet elected, in 2003, a president using mainly DREs (borrowed from TSE).

This leads us to ask if the “technical debate” over the use of VVPAT or DRE systems hold any bearings to democracy, or to the sovereignty of democratic states. If so, taking into account the U.S. Secretary of State's proclaimed mission to help spread democracy, and her pattern-fitting suggestion that Venezuela's and Argentina's are not “true democracies”³⁰, how would Mexico and Brazil fit in? What about the U.S. states that have adopted VVPAT as a norm, like Venezuela, or that mention paper ballots and ways to count them in its Constitution, like Argentina?

This question can be rephrased as one regarding the possible relations between labels for democracy and levels of sovereignty. We can take note that Argentina's government has, in 2005, called the bluff on high-risk, high-yield IMF-backed irresponsible investments that would have otherwise choked the nation's economy. That Mexico's 2006 presidential election is dealt with by U.S. mainstream media as some sort of anti-Ukraine-like story³¹. And that the DRE-elected president of Paraguay has sanctioned, in 2005, a law authorizing unlimited numbers of U.S. troops to station near his country's border with Argentina and Brazil, armed with immunity to local and international law besides guns.

Democracy can spread in different ways. Since this article aims at contributing to constructive ways, we end by stressing our view on the importance of an electronic voting system's design being consistent, as the empirical evidence raised here goes to show. For those who care for their democracies in the spirit framed here, wherever located, whatever labeled, however spread, we offer a call to beware of the rationale behind any media-driven disparagement of common voter's right to unencumbered election auditing. No amount of spinning can be a substitute for effective auditing, due to the nature of the risks involved. And for those who don't, we ask to not pretend.

References

1. Brunazo, A.: The Proconsult Case (O Caso Proconsult). In: Avaliação da Segurança da Urna Eletrônica Brasileira. Report for the III Simposium of Information Security, Brazilian Air Force Institute of Technology, São Paulo, October 2000 (2000), <http://www.brunazo.eng.br/voto-e/textos/SSI2000.htm>
2. Brunazo, A., Rezende, P.: Brazil's 2000 Senate Panel Scandal, in Stupid Security Measures for Brazil's e-Vote: Act One, Session 3. CIVILIS & Forum do Voto-E (2001), <http://www.brunazo.eng.br/voto-e/PIcontest/PI2003contest-act1.htm>

³⁰ “The Follies of Democratic Imperialism” <http://ww.worldpolicy.org/journal/articles/wpj05-sp/encarnacion.html>

³¹ Western mainstream media splashed the so-called “Orange Revolution”, in late 2004, in the aftermath of the run-off vote of the 2004 Ukrainian presidential election, taking for granted charges that it was compromised by massive corruption, voter intimidation and direct electoral fraud. Just the Opposite it did, less than two years later, with similar charges regarding the 2006 presidential election in Mexico.

3. Shelley, K.: Decertification and Whitdrawal of Approval of Accuvote-TSx Voting System, http://www.ss.ca.gov/elections/ks_dre_papers/decert.pdf
4. Schwartz, J.: High-Tech Voting System Is Banned in California, *New York Times*, May 1 (2004), <http://www.nytimes.com/2004/05/01/national/01VOTE.html>
5. Rubin, A., Kohno, T., Stubblefield, A., Wallach, D.: Analysis of an Electronic Voting System, <http://avirubin.com/vote.pdf>
6. Rezende, P.: Analysis of an Electronic Voting System (Análise de um sistema eleitoral eletrônico). *Brazilian Media Observer*, September 7 (2004), <http://observatorio.ultimosegundo.ig.com.br/artigos.asp?cod=293ENO002>
7. Rezende, P.: Analysis Review of an E-Voting System (Reavaliação da Análise de um sistema eleitoral eletrônico). *Brazilian Media Observer*, November 2 (2004), <http://observatorio.ultimosegundo.ig.com.br/artigos.asp?cod=301ENO002>
8. Rezende, P.: Electronic Voting Systems: Is Brazil ahead of its time? *Cryptobytes* 7(2), 2–8 (Fall 2004), RSA Security Labs, USA, http://www.rsasecurity.com/rsalabs/cryptobytes/CryptoBytes_Fall2004.pdf
9. Rezende, P.: Analysis of the Unicamp Report (Análise do Relatório da Unicamp. In: *Burla Eletrônica* (2002), <http://www.cic.unb.br/docentes/pedro/trabs/reunicamp.htm>
10. Rezende, P.: The Sect of the Holy Byte (A seita do Santo Byte). *Brazilian Media Observer*, September 23 (2003), <http://observatorio.ultimosegundo.ig.com.br/artigos/eno230920031.htm>
11. Rezende, P.: Diogenes' Lantern (A Lanterna de Diógenes). Report for a public hearing at Brazil's Senate (2001), <http://www.cic.unb.br/docentes/pedro/trabs/paisagem.htm>
12. Rezende, P.: Do We Need a Federal Journalism Council (Precisamos de um Conselho Federal de Jornalismo?). *Brazilian Media Observer*, September 21 (2004), <http://observatorio.ultimosegundo.ig.com.br/artigos.asp?cod=295JDB007>
13. Rezende, P.: The Anatomy of a Fraud to the Constitution (Anatomia de uma Fraude à Constituição). Dossier on the smuggling of unvoted articles into Brazil's seventh Federal Constitution, <http://www.cic.unb.br/docentes/pedro/trabs/fraudeac.html>
14. Rezende, P.: Brazil's e-vote: an overview, http://www.cic.unb.br/docentes/pedro/trabs/evote_overview.html
15. Fund, J.: *Stealing Elections: How Voter Fraud Threatens Our Democracy* Encounter Books (2004)
16. Mercuri, R.: *Electronic Vote Tabulation Checks & Balances*. Ph.D. dissertation, defended October 27, 2000 at the School of Engineering and Applied Science of the University of Pennsylvania, Philadelphia, PA (2000), <http://www.notablessoftware.com/Papers/thesdefabs.html>
17. Ansolabehere, S.: The Search for New Voting Technology. *Boston Review* (November 2001), <http://bostonreview.net/BR26.5/ansolabehere.html>

An Implementation of a Mix-Net Based Network Voting Scheme and Its Use in a Private Organization

Jun Furukawa, Kengo Mori, and Kazue Sako

NEC Corporation

j-furukawa@ay.jp.nec.com, ke-mori@bx.jp.nec.com, k-sako@ab.jp.nec.com

Abstract. We discuss an implementation of a network voting scheme based on mix-net technology. We employed the scheme presented at Financial Cryptography 2002, but replaced the numeric computations with those on an elliptic curve. As a result, we obtained three times speed up and data length shortening to one third. The system has been employed in a private organization with roughly 20,000 voters since 2004.

1 Introduction

The three basic requirements on a secure network voting system are detection of faulty voters, detection of faulty centers, and vote secrecy. Among other voting protocols achieving these three requirements, such as blind signature based schemes [FOO92, Sak94, OMAFO] and homomorphic encryption based schemes [CY85, SK94, CFSY96, CGS97, DJ01], we have chosen the mix-net based scheme [PIK93, SK95, Abe99, FS01, Ne01] for implementing our voting system.

Although the scheme requires rather large amount of computation on mixers who shuffle and decrypt the encrypted votes, and who we need to assume not to collude with all other mixers, the scheme offers many desirable features for voters and for administrators who manage voting system:

- it enjoys flexibility in representing a vote, unlike homomorphic encryption based scheme where the design of the system depends heavily on the number of choices in each vote.
- the voters can simply vote-and-go and require only a small computational ability.
- by having authorities prove the correctness of their procedures, it achieves public verifiability, that is, anyone can verify the correctness of the tally.

The cost that is paid for these properties is the computational cost to generate proofs assuring correctness of shuffling and decryption. However, it can be relaxed by elaborating computational algorithms such as use of "fixed-base comb method" and "Simultaneous multiple exponentiation method" exposed in [MOV]

As a result, the system was able to produce results that were verified correct within 6.6 minutes following a vote participated in by ten thousand voters, with three mixers each on a 1GHz PC.

For the proof, we used the scheme proposed in [EMMOS02]. Although the scheme can not prove to have zero-knowledge property, it has complete permutation-hiding property as discussed in [E04].

We note that the property of receipt-freeness is not achieved in our system. Also, the privacy of abstaining is not currently being supported. On the other hand, the administrators knowing who have voted and who have not, help to send reminders to those who have not voted.

2 The Mix-Net Based Voting Scheme

2.1 Overview

The mix-net based voting schemes achieve anonymity of votes by distributing the decryption key among multiple authorities called mixers. Voters send signed but encrypted votes. These votes will be processed by the mixers, who shuffles the votes and decrypt them. After all the mixers have done their work, the encrypted vote will be completely decrypted, but its original owner can not be identified due to the shuffles performed at every mixer. By providing shuffle-and-decrypt proofs, anyone can verify the output is the correct decryptions of the shuffled valid votes. The proofs should be made in a way it will not reveal the permutation used in the shuffle nor the decryption key, so it would not infringe the vote anonymity.

2.2 Model

We involve five kinds of players, which are

1. Election policy committee
2. Voting center
3. Shuffling management center
4. Shuffling center(mixer)
5. Voters

The election policy committee will be responsible for any fraud caused by the voting center, the shuffling management center, and the shuffling centers. The election policy committee does not engage in an actual run of the electronic voting. It takes part in determining election and security policies, and assignment of the centers. The committee authorizes the output computed by the other centers, such as the parameters determined in a set-up phase and the final tally.

The voting center is in charge of handling transactions with the voters. It will announce the voting procedures, collects pro forma votes from the authorized voters, issues receipts of the collected votes, and announces the result of the tally. The voting center will receive the result of the tally by sending the list of collected votes to the shuffling management center.

The shuffling management center is responsible for decrypting and tallying the list sent from the voting center, in collaboration with the shuffling centers(mixers). The shuffling management center passes the list to the first shuffling center, and collects his answer which will be sent to of the next shuffling

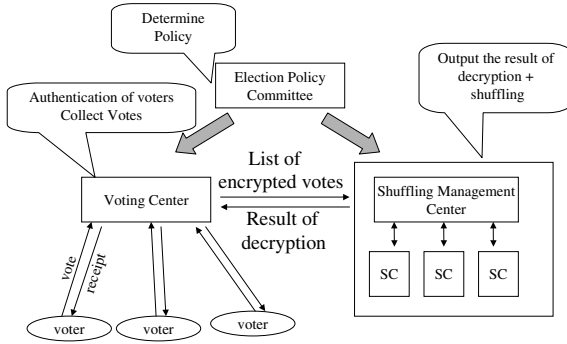


Fig. 1. System Configuration of Mix-net

center, and repeats the process until all the assigned shuffling centers shuffle and decrypt the list. The shuffled result of decryption will be sent back to the voting center. The shuffling management center is also responsible for composing a public key in the set-up phase, again in collaboration with the shuffling centers.

The shuffling center, whose other name is the mixer, is responsible for secure management of the secret key generated in the set-up phase, and conducting decryption using the key. He is also responsible for randomly shuffling the list and keeping the permutation used in a shuffle confidential.

We require for the universal verifiability, that any party can verify that all centers conducted correctly based on the policy approved by the election policy committee. Our goal in vote privacy is that it will not be infringed as long as at least one shuffling center remains honest.

Figure 1 illustrates how these players constitute an voting system. We note that the roles of the voting center and the shuffling management center can be played by one entity.

2.3 Protocol

In this subsection we describe the procedure to set-up, to encrypt votes, and to tally the votes.

In the sequel, we assume there are m shuffling centers and n voters. All the communication between the centers are digitally signed based on a public key infrastructure.

Set-Up

1. The election policy committee will determine the parameters (q, \mathbf{E}, g) which will be used in ElGamal cryptosystem on an elliptic curve \mathbf{E} . The numbers q is a prime order of the elliptic curve \mathbf{E} and g is a generator. The shuffling management center announces the authorized parameters (q, \mathbf{E}, g) to all the shuffling centers. The j -th shuffling center, SC_j , will randomly choose $x_j \text{ mod } q$ as his secret key, and report his public key $y_j = [x_j]g$

to the shuffling management center. The report is accompanied by the proof y'_j, r_j which ensures that SC_j indeed knows the secret x_j corresponding to y_j . The proof will be generated by SC_j as follows.

$$\begin{aligned} y'_j &= [\beta_j]g \\ c_j &= \mathcal{H}(p, q, g, y_j, y'_j) \\ r_j &= c_j x_j + \beta_j \pmod{q} \end{aligned}$$

with a randomly generated $\beta_j \in \mathbf{Z}/q\mathbf{Z}$.

2. The shuffling management center will verify the proof y'_j, r_j for each public key $y_j (j = 1, \dots, m)$ as follows.

$$\begin{aligned} c_j &= \mathcal{H}(p, q, g, y_j, y'_j) \\ [r_j]g - [c_j]y_j &= y'_j \\ y_j &\in \mathbf{E}, y_j \neq \mathcal{O} \end{aligned}$$

The verified public keys are combined to compose the common public key Y .

$$Y = \sum_{j=1}^m y_j$$

The proof for each public key is necessary to ensure that the common public key Y corresponds to each of the secret keys that the mixers are aware of, not those generated under a control of an adversary.

3. The election policy committee will certify the public keys y_j and Y properly generated as above.

Encryption of Votes

The $Voter_i$ will use the parameters Y and (q, \mathbf{E}, g) certified by the election policy committee and encrypt his vote m_i as follows. (We assume here that m_i is in \mathbf{E} .)

$$(G_i, M_i) = ([\bar{r}_i]g, m_i + [\bar{r}_i]Y)$$

where \bar{r}_i is an element randomly chosen by the $Voter_i$, and ID_i is information that identifies the voter. He may then prove the knowledge of m_i by generating the proof α_i, t_i by

$$\begin{aligned} \alpha_i &= [\gamma_i]g \\ c_i &= \mathcal{H}(p, q, g, Y, G_i, \alpha_i, ID_i) \\ t_i &= c_i \bar{r}_i + \gamma_i \pmod{q} \end{aligned}$$

with a randomly generated γ_i . This proof ensures that the plaintext awareness property: that is, a voter who knows the content of his vote has generated the encrypted vote. A vote duplication attack by copying someone else's encrypted vote will be thwarted here.

The voting center will verify that the voter is eligible to vote. It will verify that the proof satisfies

$$c_i = \mathcal{H}(p, q, g, Y, G_i, \alpha_i, ID_i)$$

$$[t_i]g - [c_i]G_i = \alpha_i$$

and that the elements G_i and M_i are both in \mathbf{E} . If everything is verified, then it can optionally send back a receipt of acceptance. Such a receipt cuts in two ways: it will add confidence to the voter that the center indeed accepted his vote and will be an evidence for any disputes on vote delivery. On the other hand, it will serve as a receipt in vote-buying or coercing scenario.

Tallying

The voting center will send the list of accepted votes from each voters $(G_i, M_i)_{i=1, \dots, n}$ to the shuffling management center. The shuffling management center will verify that all of each component is in \mathbf{E} , and rename them to be $(G_i, M_i) = (G_i^{(1)}, M_i^{(1)})$ for all i , which will be the input to the first shuffling center SC_1 .

The list $(G_i^{(j)}, M_i^{(j)})_i$ will be sent to SC_j . His response will be verified by the shuffling management center and will be renamed to $(G_i^{(j+1)}, M_i^{(j+1)})_i$ and sent to the next shuffling center. The response from the last shuffling center, SC_m will be verified and sent back to the voting center.

Below, we describe the procedures of each shuffling center.

1. SC_j will receive the list $(G_i^{(j)}, M_i^{(j)})_i$. He will choose a random permutation $\pi^{(j)}$ and permute the input list $(G_i^{(j)}, M_i^{(j)})_i$ and achieve the list $(\bar{G}_i^{(j)}, \bar{M}_i^{(j)})_i$ as follows:

$$(\bar{G}_i^{(j)}, \bar{M}_i^{(j)})_i = (G_{\pi^{(j)}(i)}^{(j)}, M_{\pi^{(j)}(i)}^{(j)})_i$$

2. The above permutation only changes the order of the ciphertexts, so it is easy to trace the permutation. In order to hide the permutation, we need to change the *look* of the ciphertext. The following procedure changes the look without changing the message hidden in the ciphertext.

First, SC_j combines the public keys of the subsequent shuffling centers as

$$Y_j = \sum_{\ell=j}^m y_\ell.$$

For each of $(\bar{G}_i^{(j)}, \bar{M}_i^{(j)})$, he chooses a random element $s_i^{(j)} \pmod q$ and obtains $(G'^{(j)}, M'^{(j)})_i$ by

$$G'_i{}^{(j)} = \bar{G}_i^{(j)} + [s_i^{(j)}]g$$

$$M'_i{}^{(j)} = \bar{M}_i^{(j)} + [s_i^{(j)}]Y_j$$

3. SC_j will decrypt each of $(G'_i{}^{(j)}, M'_i{}^{(j)})$ using his secret key x_j as follows:

$$M''_i{}^{(j)} = M'_i{}^{(j)} - [x_j]G'_i{}^{(j)} \quad G''_i{}^{(j)} = G'_i{}^{(j)}$$

The list $(G''_i{}^{(j)}, M''_i{}^{(j)})_i$ will be returned to the shuffling management center.

Proving Correctness

Details of procedure for mixers to prove they have correctly shuffled and decrypted the input is described in the next section.

3 Details of Correctness Proof

For simplicity, we concentrate on one shuffling center and denote his secret key as x . We represent by \bar{y} the product of the public keys of subsequent centers. What we need to prove is the correctness of the following shuffle-and-decrypt procedure.

Given n ciphertexts $(G_i, M_i)_i$, where all $\{G_i\}$ and $\{M_i\}$ are in \mathbf{E} , the shuffling center randomly chooses a permutation π and a random element $s_i \in_U \mathbf{Z}/q\mathbf{Z}$ to obtain shuffle-and-decrypt result as follows:

$$(G'_i, M'_i) = ([s_i]g + G_{\pi(i)}, [s_i]\bar{y} + M_{\pi(i)} - [x]G'_i)$$

for $i = 1, \dots, n$.

3.1 Generation of the Proof

We now provide the scheme to generate a proof that the shuffling center (which will be denoted as the prover in the sequel) indeed shuffled and decrypted honestly.

We describe the scheme in a non-interactive way, where a challenge from a verifier is given as an output of some universal one-way hash functions. We assume here that all elements of input ciphertexts (G_i, M_i) and output ciphertexts (G'_i, M'_i) are in \mathbf{E} .

To prove (G'_i, M'_i) are generated correctly from (G_i, M_i) , the prover computes the following equations for randomly chosen $z, z_i, \rho, \sigma, \tau, \lambda$ and $\lambda_i, z' \in_U \mathbf{Z}/q\mathbf{Z}$ ($i = 1, \dots, n$): We use \mathcal{H} and $\tilde{\mathcal{H}}$ to denote universal one-way hash functions which output an element of $\mathbf{Z}/q\mathbf{Z}$ and \mathbf{E} , respectively.

$$\begin{aligned} \tilde{g} &= \tilde{\mathcal{H}}(p, q, g, Y, 0), & \tilde{g}_i &= \tilde{\mathcal{H}}(p, q, g, Y, i) \\ v &= [\rho]g, & w &= [\sigma]g, & t &= [\tau]g, & u &= [\lambda]g, & u_i &= [\lambda_i]g \\ \tilde{g}'_i &= [s_i]\tilde{g} + \tilde{g}_{\pi(i)}, & \tilde{g}' &= [z]\tilde{g} + \sum_{j=1}^n [z_j]\tilde{g}_j \\ g' &= [z]g + \sum_{j=1}^n [z_j]G_j, & m' &= [z]\bar{y} + \sum_{j=1}^n [z_j]M_j \end{aligned}$$

$$\begin{aligned}
t_i &= [3z_{\pi(i)} + \tau\lambda_i]g, & \dot{v}_i &= [3z_{\pi(i)}^2 + \rho s_i]g \\
\dot{v} &= \left[\sum_{j=1}^n z_j^3 + \tau\lambda + \rho z \right]g \\
\dot{w}_i &= [2z_{\pi(i)} + \sigma s_i]g, & \dot{w} &= \left[\sum_{j=1}^n z_j^2 + \sigma z \right]g \\
c_i &= \mathcal{H}(p, q, g, \bar{y}, \tilde{g}, \{\tilde{g}_j\}, (G_j, M_j)_j, (G'_j, M'_j)_j, \\
&\quad \tilde{g}', (\tilde{g}'_j)_j, g', m', v, w, t, u, (u_j)_j, \\
&\quad (\dot{t}_j)_j, \dot{v}, (\dot{v}_j)_j, \dot{w}, (\dot{w}_j)_j, i) \\
r_i &= c_{\pi^{-1}(i)} + z_i, & r &= \sum_{j=1}^n s_j c_j + z \bmod q
\end{aligned} \tag{1}$$

$$\begin{aligned}
\lambda' &= \sum_{j=1}^n \lambda_j c_j^2 + \lambda \bmod q \\
\zeta &= \sum_{j=1}^n [c_j]G'_j, & \eta &= [x]\zeta
\end{aligned} \tag{2}$$

$$y' = [z']g, \quad \eta' = [z']\zeta \tag{3}$$

$$c' = \mathcal{H}(p, q, g, y, \zeta, \eta, y', \eta') \tag{4}$$

$$r' = c'x + z' \bmod q \tag{5}$$

The prover send the proof $g', m', \tilde{g}', \tilde{g}'_i, v, w, t, u, u_i, t_i, \dot{v}_i, \dot{v}, \dot{w}_i, \dot{w}, r, r_i, \lambda', \eta, \eta', y', r'$ ($i = 1, \dots, n$) to the verifier along with $(G'_i, M'_i)_i$.

3.2 Verifications of the Proof

The verifier first computes $(c_i)_{i=1, \dots, n}$ according to Eq.(1). Next, the verifier compute

$$\zeta = \sum_{j=1}^n [c_j]G'_j$$

and generate c' according to Eq.(4). The verifier accepts the proof if all of the following equations hold.

$$\begin{aligned}
v, t, w &\in \mathbf{E} \\
[r]g + \sum_{j=1}^n [r_j]G_j &= g' + \zeta \\
[r]\bar{y} + \sum_{j=1}^n [r_j]M_j &= \eta + m' + \sum_{j=1}^n [c_j]M'_j \\
[r]\tilde{g} + \sum_{j=1}^n [r_j]\tilde{g}_j &= \tilde{g}' + \sum_{j=1}^n [c_j]\tilde{g}'_j
\end{aligned}$$

$$\begin{aligned}
 [\lambda']g &= u + \sum_{j=1}^n [c_j^2]u_j \\
 [\lambda']t + [r]v + \left[\sum_{j=1}^n (r_j^3 - c_j^3)\right]g &= \dot{v} + \sum_{j=1}^n [c_j^2]\dot{t}_j + \sum_{j=1}^n [c_j]\dot{v}_j \\
 [r]w + [sumj(r_j^2 - c_j^2)]g &= \dot{w} + \sum_{j=1}^n [c_j]\dot{w}_j \\
 [r']g &= [c']y + y' \quad , \quad [r']\zeta = [c']\eta + \eta'.
 \end{aligned}$$

3.3 Complete Permutation Hiding

We discuss here the notion of *complete permutation hiding* (CPH) as a core requirement of unlinkability in verifiable shuffle-decryption. If a verifiable shuffle-decryption is CPH, honest verifiers will learn nothing new about its permutation from an interaction with a prover in an **overwhelming** number of cases of random tape that a prover has chosen uniformly and randomly, whereas, if the protocol is zero-knowledge, verifiers will learn nothing new in **every** case of the random tape. In other words, we define CPH so that verifiers learn nothing about the permutation in an overwhelming number of cases of common input X_n and witness W_n that the generator G_R (defined below) outputs.

Let I_n be a *set* of domain parameters $1^n, q, \mathbf{E}$, where q is prime and is of the length of the polynomial of n , and \mathbf{E} is an elliptic curve of an order q , private key \bar{x} , plain texts $\{M_i \in \mathbf{E}\}_{i=1,\dots,k}$, and random tape Z_n . Let $enc(U)$ be an *encoding of a probabilistic polynomial time (PPT) Turing machine* U which generates cipher-texts $(g_i, m_i)_{i=1,\dots,k}$ input to the shuffle-decryption procedure. We assume the existence of a knowledge extractor that can concurrently extract $\{\bar{r}_i\}_{i=1,\dots,k}$ such that $[\bar{r}_i]g_0 = g_i$ from U . This assumption is satisfied if all generators of cipher-texts are imposed to run a concurrent proof of knowledge of \bar{r}_i , and such a compulsion prevents an adaptively chosen cipher-text attack.

Definition 1. *Given $I_n(= \{1^n, q, \mathbf{E}, \bar{x} \in \mathbf{Z}/q\mathbf{Z}, \{M_i \in \mathbf{E}\}_{i=1,\dots,n}, Z_n\})$ and $enc(U)$, instance Generator G_R chooses $g_0 \in_R \mathbf{E}, x' \in_R \mathbf{Z}/q\mathbf{Z}, \{s_i \in_U \mathbf{Z}/q\mathbf{Z}\}_{i=1,\dots,k}$, and a permutation π uniformly and randomly and computes;*

$$\begin{aligned}
 m_0 &= [x' + \bar{x}]g_0, y = [x']g_0 \\
 (g_i, m_i) &= U(I_n, g_0, y) \in \mathbf{E} \times \mathbf{E} \\
 (g'_i, m'_i) &= ([s_i]g_0 + g_{\pi^{-1}(i)}, [-x']g_i + [s_i]m_0 + m_{\pi^{-1}(i)}).
 \end{aligned}$$

G_R then outputs common input X_n and witness W_n :

$$\begin{aligned}
 X_n &= \{q, \mathbf{E}, y, \bar{x}, g_0, m_0, \{(g_i, m_i)\}_{i=1,\dots,n}, \{(g'_i, m'_i)\}_{i=1,\dots,n}\}, \\
 W_n &= \{\pi, \{s_i\}_{i=1,\dots,n}, x'\}.
 \end{aligned}$$

In the above definition, U is a PPT Turing machine that plays the role of (malicious and colluding) players who generate cipher-texts $\{(g_i, m_i)\}$. Although U is

determined before the public parameter is generated, it does not lose generality because it has this public parameter as an input. In a case where U realizes honest players, it outputs

$$(g_i, m_i) = ([\bar{r}_i]g_0, M_i + [\bar{r}_i]m_0)$$

using random numbers $\{\bar{r}_i\}_{i=1,\dots,k}$ generated from the random tape Z_n .

We say X_n and W_n satisfy relation R if the following equations are satisfied:

$$\begin{aligned} m_0 &= [x' + \bar{x}]g_0, y = [x']g_0 \\ (g'_i, m'_i) &= ([s_i]g_0 + g_{\pi^{-1}(i)}, [-x']g_i + [s_i]m_0 + m_{\pi^{-1}(i)}). \end{aligned}$$

We denote this fact as $(X_n, W_n) \in R$. If there exists a witness W_n for a common input X_n that satisfies $(X_n, W_n) \in R$, common input X_n is a *correct shuffle-decryption*. Generator G_R outputs such a X_n .

Definition 2. Let $View_V^P(X_n, W_n)$ be V 's view of an interaction with P , which is composed of the common input X_n , messages V receives from P , random tape input to V , and messages V sends to P during joint computation employing X_n , where P has auxiliary input W_n s.t., $(X_n, W_n) \in R$. $View_V^P$ is an abbreviation of $View_V^P(X_n, W_n)$.

We consider the case when a semi-honest verifier may collude with malicious players who encrypt the ciphertexts and other provers who shuffle and decrypt in the same mix-net. Such a verifier and players may obtain partial information regarding the plain texts $\{M_i\}$, private key \bar{x} (the sum of other prover's private keys in the mix-net), random tapes of players, and even a part of the permutation π in addition to $View_V^P$. Moreover, they may obtain the results of other shuffle-decryptations executed by the same prover.

Then it is reasonable to describe this extra information as $H(I_n, enc(U), X_n, \pi)$ and input cipher-texts generated by the malicious player as $U(I_n, g_0, y)$ using PPT Turing machines $H(\cdot)$ and $U(\cdot)$. Note that $\{s_i\}$ are not included in the arguments of H , because we consider only the case where the prover never reveals these values to any one and the case where the prover never uses the same $\{s_i\}$ for other shuffle-decryptations.

Even though the verifier and the players may obtain the results of other shuffle-decryptations executed by the same prover who uses x' , we do not include x' into the input of U and H . Instead, we assume that there exists a PPT Turing machine K such that the distribution of $View_V^P$ for such H and U and that of $K(I_n, g_0, y, enc(U), \pi)$ are the same. We denote this as $View_V^P \approx K(I_n, g_0, y, enc(U), \pi)$. The exclusion of x' is crucial because it enables us to consider the security of shuffle-decryption over the distribution of X_n i.e., of x' .

We describe information about the permutation π that verifiers try to learn as $f(\pi)$ using PPT Turing machine f . This description can be justified because the expression $f(\pi)$ is sufficient to express any bit of π and any kind of check sum for π .

Now we can say that a verifiable shuffle-decryption protocol hides its permutations completely with respect to G_R - i.e., CPH occurs - if there exists a probabilistic polynomial time algorithm E'^E (which has black box access to E) with inputs X_n and $H(I_n, enc(U), X_n, \pi)$ that suffers no disadvantage with respect to learning anything about the permutations compared to any probabilistic polynomial time verifier E having input $View_V^P$ and $H(I_n, enc(U), X_n, \pi)$. This leads to,

Definition 3. (complete permutation hiding) *A verifiable shuffle decryption protocol (P, V, G_R) achieves complete permutation hiding if*

$$\begin{aligned} & \exists E'^E \forall E \forall H \forall f \forall U \forall c > 0 \exists N \forall n > N \forall I_n \\ & \Pr[E(View_V^P, H(I_n, enc(U), X_n, \pi)) = f(\pi)] \\ & < \Pr[E'^E(X_n, H(I_n, enc(U), X_n, \pi)) = f(\pi)] + \frac{1}{n^c}, \end{aligned} \quad (6)$$

and

$$\exists K \ View_V^P \approx K(I_n, g_0, y, enc(U), \pi)$$

where E', E, H, f, U, K are PPT Turing machine. The left probability in Eq. (6) is taken over the distribution of the random tapes input to G_R ¹ P, V, H , and E . The right probability in Eq. (6) is taken over the distribution of the random tapes input to G_R, H, E' , and E . E' may use E as a black box.

If the verifiable shuffle-decryption protocol is CPH, we can say that for **every** input ciphertexts set $\{(g_i, m_i)\}$ and its corresponding output ciphertexts set $\{(g'_i, m'_i)\}$, whatever an honest verifier who has partial information ($H(I_n, enc(U), X_n, \pi)$) about the common input (X_n), can learn about the permutation (π) after interacting with a prover, can also - in an **overwhelming** number of cases of common input (X_n)- be efficiently computed from that common input (X_n) and that partial information ($H(I_n, enc(U), X_n, \pi)$) alone using a PPT Turing machine E' without interaction with the prover as long as the prover has chosen the private key x' , permutation π , and random numbers $\{s_i\}$ uniformly and randomly.

Note that we are considering the case even where malicious and colluding players, who have the results of other shuffle-decryptations with the same x' , are engaged in generating $\{(g_i, m_i)\}$ of common input. Hence, CPH guarantees security when shuffle-decryptations with the same private key are repeatedly executed².

¹ Since the probability is taken over a distribution containing x' , we have excluded any adversary who knows x' .

² The definition of shuffle-decryption stated in [FMMOS02] is “No polynomially bounded adversary can compute any partial information of the permutation from the protocol”. Unlike our new definition, this definition does not mention the case where the verifier has already obtained partial information before the protocol begins and where the shuffle-decryptations with the same private key are repeatedly executed. These cases seem to occur quite often.

Table 1. Processing time and proof size

number of voters	Total Time		Length of Proof	
	10,000	100,000	10,000	100,000
EC Implementation	6.6 min	1 hr 7 min	4.8Mbyte	48Mbyte
Mod p Implementation ([FMMOS02])	20 min	3 hrs 44 min	12.6Mbyte	126Mbyte
Tally without proof(EC)	69 sec	12 min	–	–
Tally without proof(Mod p)	8 min	1 hrs 10 min	–	–

4 Result of Implementation

We have evaluated the system under the following conditions:

- Key Size $|q| = 160$.
- Security parameter $k=160$
- The number of shuffling centers $m = 3$.
- CPU: Pentium III 1GHz, memory 256Mbyte for each of mixers and shuffling management center.
- Communication Line 100baseTX

As a result, with a ten thousand voters the system can output a certified tally in 6.6 minutes and with a hundred thousand voters within 67 minutes, including data transmission time. Less than one-fifth of the time are required for computing the tally, and the rest of the time is devoted to proving and verifying the proof.

Table 1 compares the implementation results on Elliptic Curve and that on modular p arithmetic reported in [\[FMMOS02\]](#). Modular arithmetic requires the length of p to be 512 where as EC implementation requires 160. This affects the speed up in tallying and the shortening the proof size, both in the factor of 3.

We describe how we measured the tally. Since proving is the one that takes the most of the time, we introduced parallel scheduling. That is, a shuffling center returns the result of his shuffle-and-decrypt procedure to the shuffling management center before he starts proving the correctness of his result. The shuffling management center verifies the signature on the result and forwards the result to the next shuffling center. Thus the next shuffling center can start his job while the previous shuffling center is still engaging in the process of proving. The correctness of the result is verified by the shuffling management center as soon as the shuffling center completes generating a proof. Therefore, at a same time, one shuffling center may be engaged in a shuffle-and-decrypt procedure, another shuffling center may be proving, and a shuffling management center may be verifying the proof reported from previous shuffling center. In this parallel scheduling, we measured 'Tally without proof' when the last shuffling center reports the result of his shuffle-and-decrypt and the shuffling management center verified the signature. For 10,000 voters, it was 69 seconds after the shuffling management center send the encrypted votes to the first shuffling center. The shuffling management center had to wait another 5.5 minutes to finish receiving the proofs from three shuffling centers and verifying them each.

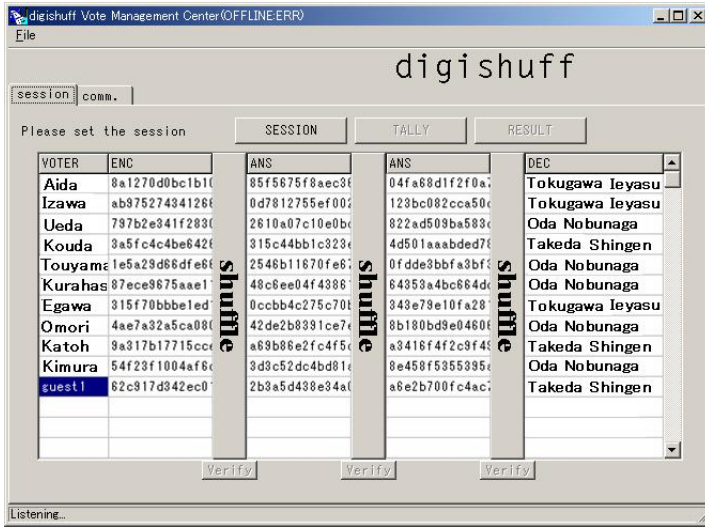


Fig. 2. An window of Tallying Result

The parallel scheduling is risky in the sense that if a shuffling center returns a properly signed wrong result, the following shuffling centers need to redo their job in order to obtain a correct tally. Redoing brings many threats. We nonetheless chose this implementation assuming such a fraud is unlikely to happen and even if it does, the shuffling center can be identified and be heavily blamed. Of course, the schedule can always be changed to sequential one, so that the next shuffling receives the input only after the input has been verified, in cases where the speed is not of a question or the threat is large. In the sequential implementation, it takes 122 seconds for a shuffling center to shuffle-and-decrypt and generate its proof for 10,000 votes, and 105 seconds for a shuffling management center to verify the proof. We also note that 67 seconds out of 122 seconds needed for a shuffling center to perform its job can be done before he receives the input.

Figure 2 shows an example of tallying result collected at Shuffling Management Center.

5 Deployment in a Private Organization

The system described above has been used in a private organization with roughly 20,000 voters since 2004. The system is being used almost every other months to make organizational decisions reflecting opinions from anonymous members of the organization. The cases include election of the president of the organization, confirming the board member, changing the by-laws, and collecting the agreements from the members to exercise organizational rights. The organization is composed of multiple divisions. All the voting was tallied within the division.

The organization had user authentication infrastructure within their intranet. After the voter authorization protocol, the voter posted the encrypted vote generated with JAVA applet.

The administrators can chose the number of mixers in each voting. The secret keys of mixers are refreshed every time. There was no major trouble after 3 years of running the system but one, where a newly assigned administrator refreshed the secret key but announced the old public key by mistake. Since the old secret key had been properly destroyed, the voting with old public key had to revoke.

It may be worth noting that the organization succeeded to cut the 90% of the cost they spend in paper-based voting. The cost cut is mostly due to the cut of man power to count paper ballots.

6 Ongoing Trial

We briefly describe here another ongoing trial using the components of the implemented system. The trial is to select young researcher award in a symposium, where roughly 650 participants of the symposium can vote among roughly 200 candidate presentations. The vote was conventionally done with paper ballots, where a voter pick the best five presentations, and write down the paper number, the title, and the presentator for each selected ones. A voter had to make a quick decision after he heard the last presentation and before he leaves the place. By introducing network voting system, it was easy for voters to search and chose the selection of his choice on web. The voters also had extra time for the selection because he can cast the ballot from his home.

Moreover, the voter is allowed to submit the vote many times, but only the latest one counts. Therefore, they could vote, say the third day of the four-days-symposium and can change the vote if there were better presentations in the last day.

7 Concluding Remarks

We discussed an implementation of mix-net based network voting scheme, which realizes many plausible features. We also discussed its actual use in binded voting within a private organization of voter size 20,000.

References

- [Abe99] Abe, M.: Mix-networks on permutation networks. In: Lam, K.-Y., Okamoto, E., Xing, C. (eds.) ASIACRYPT 1999. LNCS, vol. 1716, pp. 258–273. Springer, Heidelberg (1999)
- [Br93] Brands, S.: An Efficient Off-line Electronic Cash System Based On The Representation Problem, CWI Technical Report CS-R9323 (1993)
- [Cha81] Chaum, D.: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 84–88 (1981)
- [CY85] Cohen, J.D., Yung, M.: Distributing the power of a government to enhance the privacy of voters. In: Annual Symposium on Principles of Distributed Computing, pp. 52–62 (1985)

- [CGS97] Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. *European Transactions on Telecommunications* 8, 481–489 (1997); Preliminary version in: Fumy, W. (ed.) *EUROCRYPT 1997*. LNCS, vol. 1233, pp. 103–118. Springer, Heidelberg (1997)
- [CFSY96] Cramer, R., Franklin, M., Schoenmakers, B., Yung, M.: Secure Secret Ballot Election Schemes with Linear Work. In: *Advances in Cryptology — EUROCRYPT 1996* (1996)
- [DJ01] Damgård, I., Jurik, M.: A Generalization, a Simplification and some Applications of Paillier’s Probabilistic Public-Key system. In: *Proc. of Public Key Cryptography 2001* (2001)
- [E85] El Gamal, T.: A Public Key Cryptosystem and a Signature Scheme based on Discrete Logarithm. *IEEE Transactions on Information Theory* IT-31, 469–472 (1985)
- [EVOX] <http://theory.lcs.mit.edu/~cis/voting/voting.html>
- [FOO92] Fujioka, A., Okamoto, T., Ohta, K.: A Practical Secret Voting Scheme for Large Scale Elections. In: Zheng, Y., Seberry, J. (eds.) *AUSCRYPT 1992*. LNCS, vol. 718, pp. 244–251. Springer, Heidelberg (1993)
- [FS86] Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) *CRYPTO 1986*. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
- [F04] Furukawa, J.: Efficient, verifiable shuffle decryption and its requirement of unlinkability. In: Bao, F., Deng, R., Zhou, J. (eds.) *PKC 2004*. LNCS, vol. 2947, pp. 319–332. Springer, Heidelberg (2004)
- [FS01] Furukawa, J., Sako, K.: An Efficient Scheme for Proving a Shuffle. In: Kilian, J. (ed.) *CRYPTO 2001*. LNCS, vol. 2139, pp. 368–387. Springer, Heidelberg (2001)
- [FMOS02] Furukawa, J., Miyauchi, H., Mori, K., Obana, S., Sako, K.: An Implementation of a Universally Verifiable Electronic Voting Scheme based on Shuffling. In: Blaze, M. (ed.) *FC 2002*. LNCS, vol. 2357, pp. 16–30. Springer, Heidelberg (2003)
- [MOV] Menezes, van Oorschot, Vanstone: *Handbook of Applied Cryptography*. CRC Press, Boca Raton
- [Ne01] Neff, C.A.: A Verifiable Secret Shuffle and its Application to E-Voting. In: *ACMCCS 2001*, pp. 116–125 (2001)
- [OMAFO] Ookubo, M., Miura, F., Abe, M., Fujioka, A., Okamoto, T.: An improvement of a practical secret voting scheme. In: *Information Security Workshop 1999* (1999)
- [PIK93] Park, C., Itoh, K., Kurosawa, K.: Efficient anonymous channel and all/Nothing election scheme. In: Hellesteth, T. (ed.) *EUROCRYPT 1993*. LNCS, vol. 765, pp. 248–259. Springer, Heidelberg (1994)
- [Sak94] Sako, K.: Electronic voting schemes allowing open objection to the tally. *Transactions of IEICE E77-A* (1) (January 1994)
- [SK94] Sako, K., Kilian, J.: Secure voting using partially compatible homomorphisms. In: Desmedt, Y.G. (ed.) *CRYPTO 1994*. LNCS, vol. 839, pp. 411–424. Springer, Heidelberg (1994)
- [SK95] Sako, K., Kilian, J.: Receipt-free mix-type voting scheme. In: Guillou, L.C., Quisquater, J.-J. (eds.) *EUROCRYPT 1995*. LNCS, vol. 921, pp. 393–403. Springer, Heidelberg (1995)
- [Schnorr] Schnorr, C.P.: Efficient signature generation by smart cards. *Journal of Cryptology* 4, 161–174 (1991)
- [SENSUS] <http://lorrie.cranor.org/voting/sensus/>

The Vector-Ballot Approach for Online Voting Procedures

Aggelos Kiayias¹ and Moti Yung²

¹ Computer Science and Engineering,
University of Connecticut Storrs, CT, USA
aggelos@cse.uconn.edu

² Google Inc. and
Computer Science, Columbia University
New York, NY, USA
moti@cs.columbia.edu

Abstract. Looking at current cryptographic-based e-voting protocols, one can distinguish three basic design paradigms (or approaches): (a) Mix-Networks based, (b) Homomorphic Encryption based, and (c) Blind Signatures based. Each of the three possesses different advantages and disadvantages w.r.t. the *basic properties* of (i) efficient tallying, (ii) universal verifiability, and (iii) allowing write-in ballot capability (in addition to predetermined candidates). In fact, none of the approaches results in a scheme that simultaneously achieves all three. This is unfortunate, since the three basic properties are crucial for efficiency, integrity and versatility (flexibility), respectively. Further, one can argue that a serious business offering of voting technology should offer a flexible technology that achieves various election goals with a single user interface. This motivates our goal, which is to suggest a new “*vector-ballot*” based approach for secret-ballot e-voting that is based on three new notions: *Provably Consistent Vector Ballot Encodings*, *Shrink-and-Mix Networks* and *Punch-Hole-Vector-Ballots*. At the heart of our approach is the combination of mix networks and homomorphic encryption under a single user interface; given this, it is rather surprising that it achieves much more than any of the previous approaches for e-voting achieved in terms of the basic properties. Our approach is presented in two generic designs called “homomorphic vector-ballots with write-in votes” and “multi-candidate punch-hole vector-ballots”; both of our designs can be instantiated over any homomorphic encryption function.

1 Introduction

There are three basic paradigms for cryptographic secure ballot elections over a network. The first method is based on mix-networks where the tallying officials move the ballots between them and permute them in the process while changing their representation (e.g., partially decrypting them). Methods for doing this robustly and correctly have been designed in the last 20 years, starting with the initial work of Chaum [7]. In practical implementations, this approach in its

fully robust form (i.e., proving the correctness of the shuffling) is still considered a slow tallying process, even though there have been several steps toward more efficient designs, see e.g. [40,6,25,29,133,21]. The second method is based on homomorphic encryption (started by Benaloh [10,4,3], and then followed by many other works including [13,11,42,19,14,2]). In this general approach the ballots are encrypted and then “compressed” via a homomorphic encryption scheme into a tally. This compression property allows fast tallying, and is what makes this approach attractive. However the drawback is that pure “compressible” homomorphic encryption is not suitable to deal with write-in ballots. In fact, compression of the write-in ballots content is *not possible* since, information theoretically, if this content has no redundancy (which is always possible in the write-in ballots case) compression will ruin it. A third approach is based on blind signatures, [20], and relies on the voters obtaining a certified secret ballot from the authorities by employing a blind signature scheme [8]. This enables them to embed any form of ballot (including write-in). Subsequently, this approach requires the employment of an anonymous channel between the voter and the tallying authorities, to hide the identity of the user at the “ballot casting stage.” This requirement may be inhibiting and thus it has been suggested to combine this approach with the mix-net one so that the “anonymous channel” is implemented formally. Furthermore, while the previous paradigms support universal verifiability which assures robustness, this approach relies on taller – voter interaction and does not support it.

In recent years, there has been an increased interest in employing computer equipment for carrying out voting procedures, see e.g., [26]. In the USA this has also been motivated by the presidential election debacle in 2000 that spurred a large legislative initiative (the Help America Vote Act or HAVA) but similar initiatives materialized elsewhere in the world. One effort that took place is the joint Caltech-MIT electronic voting project. Rivest, who participated in this project, has raised the question whether it is possible to incorporate write-in ballots in homomorphic encryption based elections in a way that will still maintain its advantages and keep some of its computational gains. In fact, Rivest’s question raises the more general concern that the cryptographic paradigms optimize different goals, and business wise it may be wise to combine them under a single user interface and hope to retain some of their individual advantages and to try to gain more by a combinational approach.

Homomorphic Vector-Ballot with Write-In Votes. Motivated by this question and the issues above, we started by attacking the problem of allowing write-in ballots as follows: since homomorphic encryption elections are based on a summation register (ciphertexts are combined together which effectively sums-up the ballots under the encryption), write-in ballots need to be read individually.

To incorporate a write-in choice into a homomorphic encryption based scheme, we suggest the design of a composed ballot or a “vector ballot” that is cast by each user, and is either a regular (predetermined candidate) ballot or a write-in one with indistinguishable external representation in either case. This is the base

of the vector-ballot approach. This sounds simple, but if done in a straightforward fashion this may give voters more “free choice” in ballot representation and in cheating, and it may also give more ways to distinguish between users’ ballots. Thus, this new design leads to new concerns regarding ballot validity and ballot uniformity. In particular, it leads us to the simple yet crucial notion of *provably consistent vector ballot encodings*, which assures that in spite of the extended scenario the ballot is nevertheless legal, i.e. the voter is forced to either select a write-in, or a predetermined choice, but *not* both at the same time. Further, whenever the voter makes one of the two choices, she is forced to enter a “neutral” value in the other portion of the vector ballot. The added validation proofs by the voter makes the ballot longer, however this price (constant increase in validity proof) is reasonable given the enhancement (to be described below) it enables. The ballot representation looks the same regardless of whether the user votes for a predetermined candidate or casts a write-in ballot. After the ballot casting, the vector ballot is split into a “supposedly regular portion” and a “supposedly write-in portion” and they are processed (tallied) independently.

What we have described so far is a combination of two voting approaches: homomorphic encryption based and mix-net based. While this is important (as it allows the unification under the same user-interface of the efficient homomorphic encryption based voting with the write-in “friendly” mix-net voting), by itself the resulting scheme as a whole is not more efficient than the two individual approaches (and clearly the real bottleneck is the slow tallying robust mix-net approach).

It is thus, perhaps surprising that our approach that is based on the vector ballots has the potential to achieve more efficient tallying than any previous proposal for e-voting that allowed write-in ballots and is universally verifiable at the same time. The two major points that allow this are explained below:

1. The predetermined candidate portions of all ballots can be compressed using the efficient homomorphic encryption based tallying.
2. The write-in portions of all ballots are based on an indicator and a write-in portion. Based on such indicators we show that they can be processed using the new efficient method of *shrink-and-mix* that we propose. The method takes advantage of the fact that the vector ballots are based on homomorphic encryption and the fact that, usually, *most* of the voters select one of the predetermined candidates. Thus, using the compressibility of indicators we can eliminate a great number of unused neutral write-in portions. We note that in a fairly standard scenario, the method achieves a five-fold improvement over stand alone mix-network based election (and this will be a noticeable factor in practice, since the gain is within the system’s performance bottleneck component).

Further, the two tallying procedures above are independent. Thus, the tallying can be performed in two phases. An *on-line* phase can just perform the homomorphic encryption tallying process of the predetermined candidate portions. This is a very efficient mechanism. In most cases the actual tally and winner(s) can

be declared based on the these regular votes only and the slower tallying of the write-in portions can, in this case, be done *off-line* and at a later time. Typically, the winner will be one selected among the leading predetermined candidates of the established parties, whereas, say, Mickey Mouse (a popular write-in candidate in US elections) can afford waiting a bit longer till he knows the actual number of votes he won.

The above is the first construction within the vector-ballot approach. It shows how we achieve simultaneously the basic properties of universal verifiability and support for write-in ballots together with an efficient tallying procedure. Comparison to previous election paradigms is given in Figure 1.

Multi-Candidate Punch-Hole Vector-Ballot. A modular extension of our new approach employs our notion of *punch-hole vector ballots* which enables a more suitable scheme for voting with a large number of predetermined candidates. It extends the functionality of our vector-ballot encodings (and thus write-ins can still be incorporated). The method introduces a multitude of c summation ciphertext registers, one per candidate, while earlier schemes packed all the candidate tallies into a single summation register. Note that the vector ballot portions in the ballot design correspond to various candidates and to the corresponding summation registers. Note further that a ballot needs to have a consistent valid encoding and the voter has to prove this validity. This is different from the simplistic multi-election ballot in [4].

Employing separate registers relaxes the burden of participants by allowing them to deal with smaller ciphertexts. The gain is especially noticeable in case of many candidates. To formalize the ciphertext summation register size requirement, we introduce the notion of “capacity” of an homomorphic encryption function, which measures how many integers can be represented as plaintexts within a given summation register. Our punch-hole vector ballot design requires the capacity of the underlying encryption to be only n (the number of voters), instead of n^c required for a single summation register used previously. In fact, all leading proposed election methods in the literature that employ summation registers, [11, 19, 14], and allow for n voters and c candidates, indeed, require capacity of n^c . Note that this may cause problems in selecting the security parameter when the number of candidates is very large: e.g., if the security parameter is 1024 bits, this restricts the capacity to 2^{1024} , and if the number of candidates is large, e.g. $c = 70$, and the voting population is say around 35,000, then the capacity *cannot* contain the summation register.

An important and substantial gain in efficiency of tallying, results from the new approach when applied over the ElGamal encryption. The recovery of the final tally requires only time $O(cn)$ which is *polynomial in the number of candidates*, instead of $O(n^c)$ which is exponential in c , as it would have been the case if a straightforward brute-force has been applied for recovering the results; such steps, or slight improvements thereof employing less naive discrete-log calculations, have been employed in the past for ElGamal type of schemes in the multi-candidate setting (see, e.g., [11]). We remark that this exponential gain is

Approaches	Efficient Tallying	Univ. Verifiability	Write-ins
Homomorphic Encryption	✓	✓	×
Mix-networks	×	✓	✓
Blind-Signatures	✓	×	✓
Vector-Ballot approach	✓*	✓	✓

Fig. 1. A comparison of the current approach to previous work with respect to the following three important e-voting properties: (i) *efficient-tallying*: tallying does not require the application of a robust-mix to the total number of ballots; *in the vector ballot approach a robust-mix is still required but is applied on a *fraction* of the total number of ballots and typically as an offline operation. (ii) *universal-verifiability*: any interested third party may verify that the election protocol is executed correctly assuming a public digital record; (iii) *write-ins*: voters are allowed to enter write-in votes.

traded against a quadratic – rather than linear – (in c) work done for validity checking of ballots; in most settings this would be a reasonable price to pay for such a speedup.

A preliminary version of this work appeared in [31].

2 Preliminaries

REQUIREMENTS FOR VOTING SCHEMES. A voting-scheme needs to fulfill a variety of requirements. A brief presentation of these requirements follows.

Secrecy. Ensures the security of the contents of ballots. In the online setting, this is typically achieved by relying on the honesty of a sufficient number of the participating authorities and at the same time on some cryptographic intractability assumption. In particular, any polynomial-time probabilistic adversary that controls some arbitrary number of voters and a number of authorities (below some predetermined threshold) should be incapable of distinguishing which one of the predetermined choices a certain voter selected or whether the voter entered a write-in. In all voting schemes, once a certain number of votes have been aggregated into a partial tally, secrecy is not mandatory, e.g., once the votes of a precinct have been aggregated it is ok to reveal the partial tally (in fact in many cases it is not even desired to keep the partial tallies secret, if some regional statistics are to be extracted from the election results). Thus, voter secrecy will have an associated *Privacy Perimeter* b which will refer to the smallest number of votes that need to be aggregated into a partial tally before some information about the partial tally can be revealed; we will talk of secrecy with b -perimeter in this case.

Universal-Verifiability. Ensures that any party, including an outsider, can be convinced that all valid votes have been included in the final tally. In the online setting, where votes are casted electronically in a distributed fashion this property typically relies on the existence of a digital record that maintains the communication between all parties participated in the system. This notion was

abstracted as the “bulletin board” by Benaloh [3]. In principle one can employ Byzantine agreement, c.f. [16,17,23,22], to ensure the integrity of such a record. Moreover, in a practical implementation one can employ a publicly accessible database server for storing the bulletin board data [32] and potentially rely on database replication for maintaining the availability of the record.

Robustness. Ensures that the system can tolerate a certain number of faulty participants while it maintains its secrecy and verifiability properties. Faults may be non-malicious (e.g., processes crashing) or malicious (e.g., executing arbitrary code).

Fairness. It should be ensured that no partial results become known prior to the end of the election procedure to any subset of participants.

Another property, which we do not deal with here explicitly, is **Receipt-Freeness** [5,41,34,27,30]. Standard techniques that use re-randomizers (see e.g. [2]) can be readily employed in our schemes to allow certain forms of this property assuming the independence of ciphertext randomizing entity from coercers or malicious users.

HOMOMORPHIC ENCRYPTION SCHEMES. An encryption scheme is a triple $\langle \mathcal{K}, \mathcal{E}, \mathcal{D} \rangle$. The key-generation \mathcal{K} is a probabilistic TM which on input a parameter 1^w (which specifies the key-length) outputs a key-pair \mathbf{pk}, \mathbf{sk} (public-key and secret-key respectively). The encryption function is a probabilistic TM $\mathcal{E}_{\mathbf{pk}} : \mathbb{R} \times \mathbb{P} \rightarrow \mathbb{C}$, where \mathbb{R} is the randomness space, \mathbb{P} is the plaintext space, and \mathbb{C} the ciphertext space. When \mathbb{P} , for a given security parameter, equals \mathbb{Z}_a where a is an integer that is a function of the parameter, we will say that the encryption function has “*additive capacity*” (or just capacity) a . The correctness property of the encryption scheme is that $\mathcal{D}_{\mathbf{sk}}(\mathcal{E}_{\mathbf{sk}}(\cdot, x)) = x$ for all x independently of the coin tosses of the encryption function \mathcal{E} . If we want to specify the coin tosses of \mathcal{E} we will write $\mathcal{E}_{\mathbf{pk}}(r, x)$ to denote the ciphertext that corresponds to the plaintext x when the encryption function $\mathcal{E}_{\mathbf{pk}}$ makes the coin tosses r . Otherwise we will consider $\mathcal{E}_{\mathbf{pk}}(x)$ to be a random variable. For homomorphic encryption, we assume additionally the operations $+$, \oplus , \odot defined over the respective spaces $\mathbb{P}, \mathbb{R}, \mathbb{C}$, so that $\langle \mathbb{P}, + \rangle$, $\langle \mathbb{R}, \oplus \rangle$, $\langle \mathbb{C}, \odot \rangle$ are (families of) groups written additively (the first two) and multiplicatively respectively.

Definition 1. An encryption function \mathcal{E} is homomorphic if, for all $r_1, r_2 \in \mathbb{R}$ and all $x_1, x_2 \in \mathbb{P}$, it holds that $\mathcal{E}_{\mathbf{pk}}(r_1, x_1) \odot \mathcal{E}_{\mathbf{pk}}(r_2, x_2) = \mathcal{E}_{\mathbf{pk}}(r_1 \oplus r_2, x_1 + x_2)$.

We will consider two examples of Homomorphic Encryption schemes: “additive” ElGamal and Paillier Encryption. Both have been employed in the design of e-voting schemes in the past, see [11] and [14,2] respectively (which are also part of the current state-of-the-art schemes in the homomorphic encryption based approach). We define them below:

Additive ElGamal Encryption. It is defined by a triple $\langle \mathcal{K}, \mathcal{E}, \mathcal{D} \rangle$: the key-generation \mathcal{K} outputs the description of a finite multiplicative group \mathcal{G} of prime order q , with three generators $\langle g, h, f \rangle$ which are set to be the public-key of the system \mathbf{pk} ; the

secret-key sk is set to the value $\log_g h$. For a public-key $\langle g, h, f \rangle$ the encryption function $\mathcal{E}(r, x)$ equals the tuple $\langle g^r, h^r f^x \rangle$, and the domains $\mathbb{P} := \mathbb{Z}_q$, $\mathbb{R} := \mathbb{Z}_q$ and $\mathbb{C} := \mathcal{G} \times \mathcal{G}$ (note that we abuse the notation as $\mathbb{P}, \mathbb{C}, \mathbb{R}$ are families of sets parameterized by the security parameter). The operations $+$, \oplus are defined as addition modulo q and the operation \odot is defined as point-wise multiplication over $\mathcal{G} \times \mathcal{G}$. The decryption function \mathcal{D} for a secret-key $\log_g h$ given $\langle G, H \rangle$ it returns $H/G^{\log_g h}$, and then it performs a brute-force search over all possible values f^x to recover x . Observe that $\langle \mathbb{P}, + \rangle$, $\langle \mathbb{R}, \oplus \rangle$ and $\langle \mathbb{C}, \odot \rangle$ are all groups, and the encryption \mathcal{E} is homomorphic with respect to these operations. Finally notice that the capacity of \mathcal{E} is q (but due to the brute force required for decryption the capacity would be required to be polynomial time bounded and thus significantly less than q).

Paillier Encryption. [35]. It is a triple $\langle \mathcal{K}, \mathcal{E}, \mathcal{D} \rangle$, defined as follows: the key-generation \mathcal{K} outputs an integer N , that is a product of two safe primes, and an element $g \in \mathbb{Z}_{N^2}^*$ of order a multiple of N . The public-key of the system pk is set to $\langle g, N \rangle$ and the secret-key sk is set to the factorization of N . For a public-key $\langle g, N \rangle$, the encryption function $\mathcal{E}(r, x)$ equals the value $g^x r^N \pmod{N^2}$ and the domains $\mathbb{P} := \mathbb{Z}_N$, $\mathbb{R} := \mathbb{Z}_N^*$, and $\mathbb{C} := \mathbb{Z}_{N^2}^*$ (note that this is an abuse of notation as $\mathbb{P}, \mathbb{R}, \mathbb{C}$ are families of sets). The operation $+$ is defined as addition modulo N , and the operations \oplus, \odot are defined as multiplication modulo N^2, N respectively. The decryption function \mathcal{D} for a secret-key p, q it operates as follows: first it computes $\lambda := \lambda(N)$ the Carmichael function of N , and given a ciphertext c , it returns $L(c^\lambda \pmod{N^2})/L(g^\lambda \pmod{N^2})$ where $L(u) = \frac{u-1}{N}$ and L is defined over the set of integers $\{u \mid u \equiv 1 \pmod{N}\}$. Again, observe that $\langle \mathbb{P}, + \rangle$, $\langle \mathbb{R}, \oplus \rangle$ and $\langle \mathbb{C}, \odot \rangle$ are all groups, and the encryption \mathcal{E} is homomorphic with respect to these operations. Finally notice that the capacity of \mathcal{E} is N .

PROOFS OF KNOWLEDGE. Proofs of knowledge are protocols between two players, the Prover and the Verifier. In such protocols there is a publicly known predicate Q for which the prover knows some witness x , i.e. $Q(x) = 1$. The goal of such protocols is for the prover to convince the verifier that he indeed knows such witness. We will concentrate on “3-move” protocols for which the prover acts in the first and third move, and the verifier challenges in the second move with a random value from the proper domain (see [12]). Conversations in such protocols will be of the form $\langle a, c, r \rangle$, and the verifier will accept provided that a, c, r satisfy some conditions given as part of the specifications of the protocol. Proofs of knowledge can be made *non-interactive* by employing the Fiat-Shamir heuristics, [18] (and then, security is shown in the random-oracle model, or alternatively assuming a beacon, [39]). If the predicate Q accepts a non-interactive zero-knowledge proof, and an agent possesses a witness for Q that he wants to prove knowledge of, we will say that the agent “writes a proof for Q .” Proofs of knowledge of the above type can be combined in “AND” and “OR” fashion in an efficient manner [15],[12].

PROOFS OF KNOWLEDGE FOR HOMOMORPHIC ENCRYPTION. Let $\langle \mathcal{K}, \mathcal{E}, \mathcal{D} \rangle$ be a homomorphic encryption scheme. Below, we identify useful proof protocols (which have been used in various settings and environments).

Proof of Knowledge of Properly Formed Ciphertext for a Public Plaintext. A useful proof of knowledge in the context of e-voting is a proof that shows that a ciphertext that encrypts a publicly known plaintext is properly formed. We define the predicate $Q_{\text{cipher}}^{m,V}$ as follows $Q_{\text{cipher}}^{m,V}(r) = 1$ if and only if $\mathcal{E}_{\text{pk}}(r, m) = V$. We remark that proofs of knowledge for the two homomorphic encryptions that we consider here (additive ElGamal, and Paillier) are standard and can be done efficiently.

Proof of Knowledge for a Random Shuffle. Observe that using a homomorphic encryption scheme one can “re-randomize” a ciphertext C by computing $C' := \mathcal{E}_{\text{pk}}(0) \odot C$ (i.e., C' is uniformly distributed over all ciphertexts that correspond to the plaintext of C). Suppose now that C_1, \dots, C_k is a sequence of ciphertexts and C'_1, \dots, C'_k is a random re-encrypted permutation of these ciphertexts. We define a predicate $Q_{\text{shuffle}}^{C_1, \dots, C_k, C'_1, \dots, C'_k}$ so that $Q_{\text{shuffle}}^{C_1, \dots, C_k, C'_1, \dots, C'_k}(r_1, \dots, r_k, \pi) = 1$ if and only if $C'_{\pi(j)} = \mathcal{E}_{\text{pk}}(r_j, 0) \odot C_j$, for $j = 1, \dots, k$.

A straightforward approach for a proof for $Q_{\text{shuffle}}^{C_1, \dots, C_k, C'_1, \dots, C'_k}$ would require $\mathcal{O}(k^2)$ space. Discovering more efficient proofs is a very active area of research (as such proofs constitute the basic operation of a robust mix-network, a fundamental primitive for elections based on mixes) and several papers provided sophisticated techniques of shortening the proof as well as relaxing the robustness model to allow more efficient implementations, [25,29,33,21]. Two of the most efficient recent protocols are that of [21] and [33], that allow $\mathcal{O}(k)$ -size proofs with relatively small constant.

THRESHOLD HOMOMORPHIC ENCRYPTION SCHEMES. A (t, m) -threshold homomorphic encryption scheme is a triple $\langle \mathcal{K}, \mathcal{E}, \mathcal{D} \rangle$ so that \mathcal{K} is a protocol between a set of participants A_1, \dots, A_m , that results in the publication of the public-key pk and the sharing of the secret-key sk so that any t of them can reconstruct it. Additionally, \mathcal{D} is also a protocol between the participants A_1, \dots, A_m that results in the decryption of the given ciphertext in a publicly verifiable manner (i.e. each participant writes a proof that he follows the decryption protocol according to the specifications). Both Additive ElGamal and Paillier encryptions have threshold variants, see [37,38,24] and [19,14] respectively.

3 The Vector Ballot Approach

The participants in our schemes are the voters V_1, \dots, V_n , the authorities A_1, \dots, A_m , and the bulletin board server which is responsible for maintaining an authenticated communication transcript. Voter eligibility as well as basic ciphertext processing operations are also handled by the bulletin board server. Our voting approach is divided in four major steps: **Setup**, **Ballot-Casting**, **Tallying**, **Announcement of the Results**.

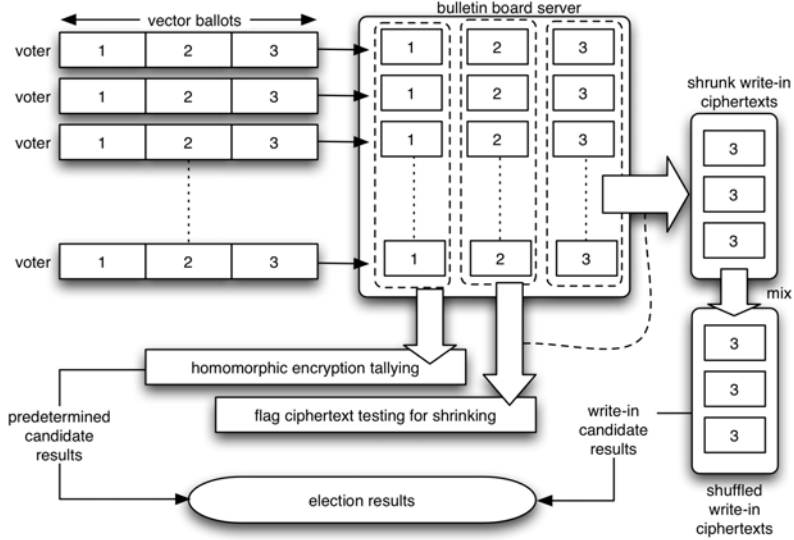


Fig. 2. The Vector-Ballot E-Voting Paradigm

In our approach, every encrypted ballot is in fact a “vector-ballot” that has three coordinates: the first is a ciphertext that contains possibly one of the predetermined election choices, the second is a flag-ciphertext that encrypts the information whether the voter selects a write-in choice or not; finally, the third coordinate possibly contains a write-in choice. A proof of “consistent ballot encoding” will be broken into a number of “consistency arguments” and will ensure that the vector ballot is formed properly (i.e., it either contains a predetermined choice in the first coordinate or a write-in choice in the last coordinate and furthermore the “flag” value is encrypted consistently). The tallying phase has two independent phases: (i) tallying the non-write-in election results using the homomorphic encryption function properties; (iia) shrinking the number of write-in votes using the flag-ciphertexts; (iib) employing a mix-net over the shrunken write-in ballot sequence. The general overview of these procedures is presented in Figure 2. We describe our approach in detail in the following subsections.

3.1 Setup and Capacity Assumption

In our approach we will employ a threshold homomorphic encryption function $\langle \mathcal{K}, \mathcal{E}, \mathcal{D} \rangle$. We will also employ the necessary assumption regarding the capacity of the encryption function:

Assumption 1. Capacity Assumption. *The capacity of the encryption function satisfies $a > M^c$ where c is the number of candidates, and M an integer with $M > n$ (the number of voters).*

Setup. The authorities A_1, \dots, A_m execute the protocol \mathcal{K} which results in the publication in the bulletin board of the public-key \mathbf{pk} . At the same time the secret-key \mathbf{sk} is shared amongst the authorities A_1, \dots, A_m .

3.2 Ballot-Casting Step

Each eligible voter gets authorized to the bulletin board and reads the public-key \mathbf{pk} of the system. The set of choices is defined as $\text{Choices} := \{1, M, M^2, \dots, M^{c-1}\}$ where M is an integer with the property $M > n$.

Forming the Vector-Ballot. Each voter V_i publishes a vector ballot $\langle C_1[i], C_2[i], C_3[i] \rangle$. If the voter wishes to select one of the predetermined choices of the election she selects $C_1[i]$ to be an encryption of one of the values in the set Choices while $C_2[i], C_3[i]$ are encryptions of 0; in particular $C_1[i] := \mathcal{E}_{\mathbf{pk}}(M^{\ell_i-1})$ where $\ell_i \in \{1, \dots, c\}$ is the personal choice of the voter, and $C_2[i] := \mathcal{E}_{\mathbf{pk}}(0), C_3[i] := \mathcal{E}_{\mathbf{pk}}(0)$. If the voter wishes to enter a write-in ballot she selects $C_1[i]$ to be an encryption of 0, $C_2[i]$ to be an encryption of 1, and $C_3[i]$ to be an encryption of some string string_i which is the voter's write-in entry. Formally, $C_1[i] := \mathcal{E}_{\mathbf{pk}}(0), C_2[i] := \mathcal{E}_{\mathbf{pk}}(1)$ and $C_3[i] := \mathcal{E}_{\mathbf{pk}}(\text{string}_i)$. Together with her vector ballot the voter must publish a proof of “consistent ballot encoding.” In particular V_i writes a proof for the following predicate:

$$\left((Q_{\text{cipher}}^{1, C_1[i]} \vee Q_{\text{cipher}}^{M, C_1[i]} \vee \dots \vee Q_{\text{cipher}}^{M^{c-1}, C_1[i]}) \wedge Q_{\text{cipher}}^{0, C_2[i]} \wedge Q_{\text{cipher}}^{0, C_3[i]} \right) \vee (Q_{\text{cipher}}^{0, C_1[i]} \wedge Q_{\text{cipher}}^{1, C_2[i]})$$

The above proof can be done *efficiently* as discussed in section 2, since it is an AND/OR composition of the proof of knowledge for the predicate $Q_{\text{cipher}}^{m, V}$ which can be done quite efficiently for either of the two homomorphic encryption functions that we consider. Moreover it only adds a constant overhead compared to proofs of previous homomorphic-encryption based voting schemes.

Regarding the above proof of consistent ballot encoding it is easy to prove the following fact:

Fact 1. *The only ballot encodings for $\langle C_1[i], C_2[i], C_3[i] \rangle$ allowed are:*

- (i) *The second ciphertext encrypts a 0, the first ciphertext contains a value from the set Choices and the third ciphertext encrypts a 0.*
- (ii) *The second ciphertext encrypts a 1, the first ciphertext encrypts a 0 and the third ciphertext is unrestricted.*

In the end of the ballot-casting step the bulletin board authority may seal the election, by signing the contents of the bulletin-board.

3.3 Tallying Step

The Non-write-in Part. The vector ballots are parsed so that the first component is collected and the sequence of ciphertexts $C_1[1], \dots, C_1[n]$ is formed. The “tally ciphertext” is defined as $C_{\text{tally}} = C_1[1] \odot \dots \odot C_1[n]$. It is easy to see that due to the homomorphic property and the capacity assumption, C_{tally} is a

ciphertext that hides a value T that satisfies $T = \sum_{i \in V} M^{\ell_i - 1}$ (as an integer), where $V \subseteq \{1, \dots, n\}$ is the set of voters that did not select the write-in option. Observe that if k_0, \dots, k_{c-1} are the tallies won by each of the c candidates, it holds that $T = k_0 + k_1 M + \dots + k_{c-1} M^{c-1}$, and $k_0, \dots, k_{c-1} < M$, i.e., if we write T as an integer in base M we can obtain the counts for each candidate.

Dealing with the Write-ins — Shrink-and-Mix networks. Write-in ballots are not “compressible” into a single ciphertext like regular ballots and thus they have to be mixed and revealed one by one. Nevertheless our approach allows for a significant efficiency improvement that we call a **Shrink-and-Mix** network. A shrink-and-mix network for voting is a mix-network that attempts to shrink the input ballot sequence prior to the mix procedure in order to gain efficiency. (Indeed gaining efficiency in settings where it is possible is crucial, given the state of the art of Mix networks, see [25]). Shrink-and-mix is a concept that naturally binds to our approach that combines write-in ballots with regular homomorphic encryption based e-voting. This is because in our approach the following unique properties are true:

1. Most voters will not cast a write-in ballot, but rather select one of the predetermined choices of the election.
2. There is a way to employ the homomorphic properties of the encryption function to test whether a small batch of encrypted vector ballots contains a write-in without violating the privacy of the voters (given security perimeter b).
3. There is a way to find the exact number of write-in votes prior to opening them, without violating the secrecy of the voters (given security perimeter b).

Justification. For item 1, observe that in most settings the write-in option will be used sparingly by the voters who will typically select one of the predetermined candidates for the election. For item 2, recall that in the vector-ballot approach, each vector ballot $\langle C_1[i], C_2[i], C_3[i] \rangle$ contains a “flag-ciphertext” (the value $C_2[i]$) that encrypts the value 0 or 1 depending on whether the voter voted with one of the predetermined choices (in $C_1[i]$) or entered a write-in (in $C_3[i]$). Suppose now that we have a set of voters i_1, \dots, i_b and we want the authorities to check whether one of them entered a write-in without violating the privacy of the voters. Then, simply the authorities collect the flag ciphertexts from the vector ballots of these voters $C_2[i_1], \dots, C_2[i_b]$ and decrypt the ciphertext $C_{i_1, \dots, i_b} := C_2[i_1] \odot \dots \odot C_2[i_b]$. Now observe that the decryption of C_{i_1, \dots, i_b} is the number of write-in votes entered by the voters $\{i_1, \dots, i_b\}$ and thus the authorities are capable of deducing whether there is a write-in entry among the ciphertexts $\{C_3[i_1], \dots, C_3[i_b]\}$. We note that it is possible to further enhance the privacy preserving aspects of this step by employing a protocol that instead of decrypting C_{i_1, \dots, i_b} it merely tests whether the ciphertexts decrypts to 0 or not. Still in many settings the privacy leakage may not be significant (given that the privacy perimeter b is sufficiently large).

For item 3, observe that the ciphertext $C_{1, \dots, n} := C_2[1] \odot \dots \odot C_2[n]$ is an encryption of the number of write-in votes. Thus if the authorities wish to find efficiently the exact number of write-ins they have to compute $C_{1, \dots, n}$ and decrypt

it (recall that linear computations in the number of voters constitute practically optimal complexity for the tallying phase).

Given the above properties we can now describe the shrink-and-mix method which is divided in two separate stages (perhaps not surprisingly) named: (a) shrink and (b) mix.

Shrink Stage. First we describe the shrink stage in detail below:

- **Input:** the sequence of all vector-ballots. Let $V \subseteq \{1, \dots, n\}$ be the subset of voters that entered a non-write-in ballot and denote by V' the set $\{1, \dots, n\} - V$ (the subset of voters that entered a write-in).
- **Output:** a set V^* such that $V' \subseteq V^* \subseteq \{1, \dots, n\}$.
- **Initialize.** The authorities A_1, \dots, A_m compute the number of write-ins h (feasible by item 3 above). Let p denote the probability that an arbitrary voter enters a write-in defined as $p := h/n$. Let b be the desired privacy perimeter for the elections.
- **Shrink.** Let $\sigma := \langle C_2[1], \dots, C_2[n] \rangle$ be the sequence of the second components of all ballot vectors and let V^* initially defined to $\{1, \dots, n\}$. The authorities divide σ into n/b batches so that each batch contains b ciphertexts. Since the probability of an arbitrary voter to enter a write-in is p it follows that the probability that a batch contains no write-in is $(1-p)^b$. The authorities test whether each one of the n/b batches contains a write-in or not (as described in the item 2 above). If the batch of flag-ciphertexts that corresponds to the voters $\{i_1, \dots, i_b\}$ does not contain a write-in we modify $V^* = V^* - \{i_1, \dots, i_b\}$. Assuming that each batch is independent from the other, it follows that the expected number of batches without a write-in is $\frac{n}{b}(1-p)^b$, so the expected size of V^* will be $n - n(1-p)^b$. Observe that the correctness of the shrink stage (i.e. $V' \subseteq V^* \subseteq \{1, \dots, n\}$) follows easily. The closeness of V^* to V' (note that $|V^*| - |V'| = n(1-p)(1 - (1-p)^{b-1})$) can be calibrated by lowering the parameter b (at the expense of reducing privacy).

Mix Stage. The mix-stage is described next.

- **Input:** A sequence of ciphertexts $\sigma^* := \langle G[i] \rangle_{i=1, \dots, n^*}$, where $n^* = |V^*|$, V^* is the output of the shrink stage and $\langle G[1], \dots, G[n^*] \rangle = \langle C_3[i] \mid i \in V^* \rangle$.
- **Output:** a sequence of ciphertexts $\langle G'[i] \rangle_{i=1, \dots, n^*}$ so that there is a permutation π on $\{1, \dots, n^*\}$ that satisfies $G'[i]$ is a random re-encryption of $G[\pi(i)]$.
- **Mix.** The authorities A_1, \dots, A_m execute a “robust mix” for the sequence of ciphertexts $\sigma^* = \langle G[1], \dots, G[n^*] \rangle$. This can be accomplished by employing any existing robust-mix method, [25,29,33,21]. The most straightforward robust mix technique has each authority re-encrypting each ciphertext $G[i]$ and permuting the whole sequence randomly to obtain the sequence $\langle G'[1], \dots, G'[n^*] \rangle$ and also writing a proof for $Q_{\text{shuffle}}^{G[1], \dots, G[n^*], G'[1], \dots, G'[n^*]}$. Note that authorities perform the above steps in sequence by acting on the output of the previous authority.

We remark that robust mixes are expensive in terms of computation and space; for this reason the shrink stage that our model allows can be crucial for the improvement of the efficiency of the mixing. The *shrink ratio* for a shrink-and-mix network is the expected reduction percentage of the given sequence of ciphertexts $\langle C_3[i] \rangle_{i \in V^*}$ i.e., the fraction $(n - |V^*|)/n$. Observe that it holds that $(n - |V^*|)/n = (1 - p)^b$ and thus the expected shrink ratio equals $(1 - p)^b$. To illustrate the gain we obtain using the shrink-and-mix network consider the following scenario: in many elections it is reasonable to expect a write-in probability $1/100$, so by setting the privacy perimeter $b = 20$ (which can be reasonable for the privacy of the voters in most settings – especially given that only the fact “a write-in exists in a batch of b votes” will be revealed) we obtain a shrink ratio of approximately 0.81, which means that 81% of the ciphertexts will be discarded prior to the execution of the robust mix. This translates to a significant gain in the efficiency of the mixing procedure.

3.4 Announcement of the Results

First the authorities announce the results for the non-write-in part of the election (in fact, this step can be performed prior to the execution of the shrink-and-mix network). The authorities A_1, \dots, A_m execute the verifiable decryption protocol \mathcal{D} on the ciphertext C_{tally} to reveal the value T . Due to the properties of the value T it holds that if T , as an integer, is written in base M , then the tallies for each candidate are revealed (cf. section 3.3); note that due to the capacity assumption there will be no wrap-arounds during the computation of the tally ciphertext C_{tally} .

Subsequently the authorities execute the shrink-and-mix network (and frequently this will be done after the winner of the election is already determined from the non-write-in votes) and then they execute the protocol \mathcal{D} for each of the ciphertexts $G_*[1], \dots, G_*[n^*]$ that belong to the output of the shrink-and-mix network. This will reveal all the strings string_i for $i \in V^*$, where V^* is the output of the shrink stage. Since $V' \subseteq V^* \subseteq \{1, \dots, n\}$ (recall that V' is the subset of voters that actually chose a write-in option) all entered write-ins will be revealed (with, perhaps, a number of 0's that correspond to the ciphertexts that were entered by the voters in $V^* - V'$). When $\text{string}_i = 0$ the entry will be removed from the write-in vote listing (recall that “0” is not considered a valid write-in vote). The final elections results consist of the counts for each of the pre-determined candidates as well as counts for the write-in selections.

3.5 Properties of the Paradigm

Efficiency. First note that our vector-ballot approach can be readily instantiated over the two homomorphic encryption functions (additive ElGamal or Paillier) that we describe in section 2.

The Voters' Perspective. The activity of each voter in the vector-ballot approach includes the following operations: after the setup phase each voter must be authenticated to the bulletin board server. The bulletin board server maintains

the listing with all eligible voters. After authentication, the voter reads from the bulletin board the public-key of the authorities and all other information that is pertinent to the election, i.e., the listing of predetermined candidates. The voter privately decides on one of the predetermined candidates or to a certain write-in choice and publishes her encrypted ballot which consists of the three ciphertexts as described in section 3.2. Further she needs to publish the proof of consistent ballot-encoding. This is done by writing in the bulletin board the non-interactive zero-knowledge proof as described in section 3.2. This proof has size linear to the number of predetermined candidates and can be generated very efficiently for the two homomorphic encryption schemes that we consider.

The Authorities' Perspective. The work of the authorities is divided in two separate stages. (i) Before ballot-casting the authorities execute the **Setup** stage of the election that requires them to run the key-generation protocol of the employed threshold homomorphic encryption scheme. (ii) After the ballot-casting phase the authorities proceed to the tallying phase. The aggregation of the non-writein part of voters' encrypted ballots is a linear operation in the number of voters that employs the homomorphic property of the underlying encryption scheme. Observe that this task can be arbitrarily distributed to any number of entities. Given the aggregated ciphertext, the authorities decrypt it by executing the decryption protocol of the underlying homomorphic encryption scheme; this reveals the counts for the predetermined candidates. It is highly likely that a winner of the election can be already determined at this stage. Subsequently, the authorities execute the shrink-and-mix protocol. This requires the authorities to execute a robust-mix protocol, but *only* over the encrypted writein ballots that remain after the shrinking phase. The shrinking phase by itself is efficient as it is only linear in the number of encrypted ballots. Subsequently the execution of the robust-mix is performed in the shrunk writein encrypted ballot sequence which may allow a significant gain as it is argued in section 3.3. Furthermore any robust mix can be used in a black-box fashion by our shrink-and-mix method; thus we can take advantage of any sophisticated robust shuffling protocol, e.g. the schemes of [25,29,33,21].

Comparison to Previous Approaches. We first observe that the efficiency of our scheme is comparable to previous approaches in the homomorphic encryption based election. In fact the only difference is the small constant overhead that is introduced in the part of the voter since she has to provide a proof of a consistent ballot encoding. In previous homomorphic encryption based solutions the “proof of ballot-validity”, is also linear in the number of candidates; note that this cannot be improved further if we use encrypted-ballots coupled with a “1-out-of- c ” non-interactive zero-knowledge proof (which has by definition length linear in c). Going beyond the homomorphic-encryption approach, our approach allows the incorporation of writein votes. In this respect, we first observe that our schemes achieve universal-verifiability, unlike the previous writein approach based on blind signatures. When compared to the mix-network approach, we also employ a “robust-mix” but we do so with a significant gain compared to

the previous mix network protocols: indeed since the great majority of the voters will not cast a write-in vote, our *shrink-and-mix* approach will achieve, e.g., a five-fold improvement (assuming write-in probability of $1/100$, see section 3.3). This can be a significant improvement in a practical setting.

Security. Regarding the security properties of our scheme we make the following claim: The e-voting approach described above satisfies secrecy with b -perimeter, universal-verifiability, robustness and fairness provided that (i) less than t authorities are malicious (where t is the threshold required for performing decryption), (ii) the underlying homomorphic encryption scheme is semantically secure, (iii) participants can consult a beacon for the purpose of generating challenges for the zero-knowledge proofs.

Justification. First we argue about Universal-verifiability: a third party auditor can verify that all votes have been counted by performing the following three steps: (i) *verifying the non-write-in part*: the auditor recomputes the tally ciphertext C_{tally} from the first portion of every voter's vector-ballot and verifies that the authorities decrypted C_{tally} properly by checking the non-interactive zero-knowledge proof of decryption; (ii) *verifying the shrinking phase*: the auditor recomputes all ciphertexts C_{i_1, \dots, i_b} that were used in the shrinking stage of the shrink-and-mix network and verifies their decryption as in (i). (iii) *verifying the robust mix*: the auditor checks all mixing proofs given by the shuffling authorities during the mixing procedure. Regarding fairness, we observe that no partial sum can be revealed to any third party due to the semantic-security of the homomorphic encryption function and the zero-knowledge properties of the proofs of consistent ballot encodings. Regarding robustness observe that it is guaranteed unconditionally for voters: any eligible voter may fail without having any impact on the protocol; furthermore, any number of authorities below the threshold t may fail without affecting the protocol. Note that we do not deal with failures explicitly affecting the bulletin board server which is a formalism that we use in a black-box fashion. Finally, the secrecy with b -perimeter of our scheme is justified based on the semantic security of the underlying homomorphic encryption scheme.

4 Punch-Hole / Write-in Ballots

In settings where the number of candidates c and the number of voters n is large it could be the case that it might be detrimental (in terms of efficiency) to use any scheme based on the homomorphic encryption approach ([11, 19, 14]) as well as our approach of the previous section. This is because the capacity assumption (employed by all the above protocols) mandates that the capacity a of the encryption function satisfies the condition $a > n^c$. Even worse if the additive ElGamal instantiation is used (as e.g. in the case of the scheme of [11]) the tallying phase would require a brute-force step proportional to n^c which is very expensive (or $n^{c/2}$ using a baby-step giant-step time-space tradeoff). For

such cases we introduce an alternative generic vector ballot design for our e-voting approach that is capable of dealing with such settings very efficiently. In the variant of our approach of this section, the ballot of each voter consists of $c + 2$ ciphertexts (instead of 3) and the only allowed ballot encodings are the following (i) encrypt a single “1” in the first c ciphertexts and “0” everywhere else, or (ii) enter a write-in ballot in the last ciphertext, encrypt a “0” in the first c ciphertexts and encrypt a “1” in the $(c + 1)$ -th ciphertext (which plays the role of the “flag-ciphertext”). The encoding can be thought of as “punch-hole/write-in” voting because the voter either “punches” a hole in the first c locations (by voting “1”) or enters his write-in choice in the last location. In the remaining we briefly explain the approach, mentioning only the cases where there is significant difference from our paradigm of section 3.

First we note that the capacity assumption will be relaxed as follows:

Assumption 2. Relaxed Capacity Assumption. *The capacity a of the encryption function satisfies $a > n$ (the number of voters).*

Forming the Vector-Ballot. Each voter V_i publishes a vector ballot $\langle C_1[i], C_2[i], \dots, C_{c+1}[i], C_{c+2}[i] \rangle$. If the voter wishes to select one of the predetermined choices $\{1, \dots, c\}$ of the election she selects $C_{\ell_i}[i] := \mathcal{E}_{\text{pk}}(1)$, where $\ell_i \in \{1, \dots, c\}$ is her choice, and then sets $C_\ell[i] := \mathcal{E}_{\text{pk}}(0)$ for all $\ell \in \{1, \dots, c + 2\} - \{\ell_i\}$. On the other hand, if the voter wishes to enter a write-in she selects $C_{c+2}[i] := \mathcal{E}_{\text{pk}}(\text{string}_i)$ where string_i is her write-in choice, and sets $C_{c+1}[i] := \mathcal{E}_{\text{pk}}(1)$ as well as $C_\ell[i] := \mathcal{E}_{\text{pk}}(0)$ for $\ell = 1, \dots, c$. Together with her vector ballot the voter publishes a proof of a consistent vector ballot encoding to ensure that her ballot is formed properly. More specifically this is done as follows:

(Consistency Argument #1). V_i shows that the first $c + 1$ locations of her vector ballot contain only a single 1 among c 0’s; this is accomplished as follows: for each $\ell = 1, \dots, c + 1$, V_i produces a proof for the predicate $(Q_{\text{cipher}}^{0, C_\ell[i]} \vee Q_{\text{cipher}}^{1, C_\ell[i]})$, i.e., showing that the $C_\ell[i]$ ciphertext either encrypts a 0 or a 1. Then she calculates the ciphertext $C_{\text{agg}}[i] = C_1[i] \odot \dots \odot C_{c+1}[i]$ and produces a proof for the predicate $Q_{\text{cipher}}^{1, C_{\text{agg}}[i]}$, i.e., she shows that it is an encryption of 1.

(Consistency Argument #2). The voter shows that either the two last ciphertexts in the vector ballot encrypt 0, or that the $(c + 1)$ -th ciphertext encrypts a 1, i.e., V_i produces a proof for the predicate $(Q_{\text{cipher}}^{0, C_{c+1}[i]} \wedge Q_{\text{cipher}}^{0, C_{c+2}[i]}) \vee (Q_{\text{cipher}}^{1, C_{c+1}[i]})$.

Alltogether the voter will have to show the following predicate for ballot consistency:

$$\bigwedge_{\ell=1}^{c+1} (Q_{\text{cipher}}^{0, C_\ell[i]} \vee Q_{\text{cipher}}^{1, C_\ell[i]}) \wedge Q_{\text{cipher}}^{1, C_{\text{agg}}[i]} \wedge \left((Q_{\text{cipher}}^{0, C_{c+1}[i]} \wedge Q_{\text{cipher}}^{0, C_{c+2}[i]}) \vee (Q_{\text{cipher}}^{1, C_{c+1}[i]}) \right)$$

It is easy to verify that the above consistency arguments enforce the intended ballot-encodings as stated in the following fact:

Fact 2. *The only feasible ballot encodings allowed by the consistency arguments above are:*

- (i) The $(c + 1)$ -th flag-ciphertext encrypts a 0, the first c ciphertexts contain a single 1 among $c - 1$ zero's and the $(c + 2)$ -th ciphertext encrypts a 0.
- (ii) The $(c + 1)$ -th flag-ciphertext encrypts a 1, the first c ciphertexts all encrypt 0's, and the $(c + 2)$ -th ciphertext is unrestricted.

In the end of the ballot-casting step the bulletin board authority may seal the election, by signing the contents of the bulletin-board.

4.1 Tallying Step

The Non-write-in Part. In the tallying phase, the vector ballots are parsed so that the first c components are collected and the c sequences of ciphertexts $C_\ell[1], \dots, C_\ell[n]$ are formed for $\ell = 1, \dots, c$. We define c “tally ciphertexts” as $C_{\text{tally}}^\ell = C_\ell[1] \odot \dots \odot C_\ell[n]$. It is easy to see that due to the homomorphic property, C_{tally}^ℓ is a ciphertext that hides a integer value T_ℓ that equals the number of votes that were won by the predetermined election candidate $\ell \in \{1, \dots, c\}$. Decrypting these ciphertexts reveals the votes accumulated by each predetermined candidate. Dealing with the write-in part of each vector-ballot is as in the paradigm of section 3.

Security and Efficiency. The security of the punch-hole/write-in version of our paradigm can be argued in similar terms as the main paradigm. Regarding efficiency, the main difference between the punch-hole paradigm and the general vector-ballot paradigm is that the encrypted ballot contains $c + 2$ ciphertexts instead of 3. While this may sound as a substantial increase in space it is not necessarily so: indeed, the security parameter in the vector-ballot paradigm (as well as in any homomorphic encryption scheme) is lower bounded by $c \cdot \log n$, whereas the security parameter in the punch-hole approach is independent of c (and is lower bounded by $\log n$). Thus the two approaches are not substantially different in terms of space when c and n are very large. In terms of time-efficiency, the punch-hole approach requires more work from the voter in the proof of the vector-ballot consistency, but it yields a significant gain from the fact that the security parameter does not have to be proportional to the number of candidates and that tallying (as described below) can be done very efficiently over additive ElGamal encryption — in fact, an exponential gain.

Exponential gain for the additive-ElGamal instantiation. Observe that when the above protocol approach is instantiated with additive ElGamal encryption the announcement of the results requires c brute-force searches of a space of size n instead of a brute-force step of a space of size n^{c-1} as it is the case with previous ElGamal-based encryption-schemes (e.g. [11]). This emphasizes further the usefulness of the “punch-hole” approach to increase the efficiency of the system. We remark that this significant gain is independent of the addition of the writein part of the election and in fact it can be also executed in the non-writein setting of [11].

Remark. It has been brought to our attention that a scheme related to the punch-hole approach (without the combination of vector-ballots/ write-in votes) appeared in the Ph.D. Thesis of M. Hirt [28].

Space-Time Tradeoffs. It is possible to obtain a hybrid between the standard paradigm and the punch-hole by having a vector of registers of smaller capacity that will be aggregating the votes for small subsets of candidates (as opposed to a single candidate in the punch-hole case above). This yields an immediate space-time tradeoff in terms of encrypted ballot encoding and time to recover the result.

Acknowledgement. We thank Ron Rivest for his motivating question regarding homomorphic encryption and writein ballot combination and to Pierre-Alain Fouque for getting it to our attention and for related discussions.

References

1. Abe, M., Hoshino, F.: Remarks on Mix-Networks Based on Permutation Networks. In: PKC 2001 (2001)
2. Baudron, O., Fouque, P.-A., Pointcheval, D., Poupard, G., Stern, J.: Practical Multi-Candidate Election system. In: The Proceedings of the ACM Symposium on Principles of Distributed Computing PODC (2001)
3. Benaloh, J.: Verifiable Secret-Ballot Elections, PhD Thesis, Yale University (1987)
4. Benaloh, J., Yung, M.: Distributing the Power of a Government to Enhance the Privacy of Voters. In: The proceedings of the ACM Symposium on Principles of Distributed Computing PODC (1986)
5. Benaloh, J., Tuinstra, D.: Receipt-Free Secret-Ballot Elections. In: STOC 1994 (1994)
6. Boneh, D., Golle, P.: Almost Entirely Correct Mixing With Applications to Voting. In: 9th ACM-CCS Conference (2002)
7. Chaum, D.: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM* 24(2), 84–88 (1981)
8. Chaum, D.: Blind Signatures for Untraceable Payments. In: CRYPTO 1982, pp. 199–203 (1982)
9. Chaum, D.: Elections with unconditionally-secret ballots and disruption equivalent to breaking RSA. In: Günther, C.G. (ed.) EUROCRYPT 1988. LNCS, vol. 330, pp. 177–182. Springer, Heidelberg (1988)
10. Cohen (Benaloh), J.D., Fischer, M.J.: A Robust and Verifiable Cryptographically Secure Election Scheme. In: FOCS 1985 (1985)
11. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 103–118. Springer, Heidelberg (1997)
12. Cramer, R., Damgård, I.B., Schoenmakers, B.: Proof of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)
13. Cramer, R., Franklin, M.K., Schoenmakers, B., Yung, M.: Multi-authority secret-ballot elections with linear work. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 72–83. Springer, Heidelberg (1996)

14. Damgård, I., Jurik, M.: A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer, Heidelberg (2001)
15. De Santis, A., Di Crescenzo, G., Persiano, G., Yung, M.: On Monotone Formula Closure of SZK. In: FOCS 1994 (1994)
16. Feldman, P., Micali, S.: Byzantine agreement in constant expected time (and trusting no one). In: 26th Annual Symposium on Foundations of Computer Science (FOCS '85), Los Angeles, Ca., USA, October 1985, pp. 267–276. IEEE Computer Society Press, Los Alamitos (1985)
17. Feldman, P., Micali, S.: Optimal algorithms for byzantine agreement. In: Cole, R. (ed.) Proceedings of the 20th Annual ACM Symposium on the Theory of Computing, Chicago, IL, May 1988, pp. 148–161. ACM, New York (1988)
18. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
19. Fouque, P.-A., Poupard, G., Stern, J.: Sharing decryption in the context of voting or lotteries. In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, p. 90. Springer, Heidelberg (2001)
20. Fujioka, A., Okamoto, T., Ohta, K.: A Practical Secret Voting Scheme for Large Scale Elections. In: ASIACRYPT 1992 (1992)
21. Furukawa, J., Sako, K.: An efficient scheme for proving a shuffle. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 368–387. Springer, Heidelberg (2001)
22. Galil, Z., Mayer, A., Yung, M.: Resolving message complexity of byzantine agreement and beyond. In: Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science, FOCS 1995, October 23–25, pp. 724–733. IEEE Computer Society Press, Los Alamitos (1995)
23. Garay, J., Moses, Y.: Fully polynomial byzantine agreement in $t + 1$ rounds. In: Aggarwal, A. (ed.) Proceedings of the 25th Annual ACM Symposium on the Theory of Computing, San Diego, CA, USA, May 1993, pp. 31–41. ACM Press, New York (1993)
24. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, p. 295. Springer, Heidelberg (1999)
25. Golle, P., Zhong, S., Boneh, D., Jakobsson, M., Juels, A.: Optimistic mixing for exit-polls. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 451–465. Springer, Heidelberg (2002)
26. Gritzalis, D. (ed.): Secure Electronic Voting, Advances in Information Security, vol. 7. Kluwer, Dordrecht (2002)
27. Hirt, M., Sako, K.: Efficient receipt-free voting based on homomorphic encryption. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, p. 539. Springer, Heidelberg (2000)
28. Hirt, M.: Multi-Party Computation: Efficient Protocols, General Adversaries, and Voting, Ph.D. Thesis, ETH Zurich (2001)
29. Jakobsson, M., Juels, A., Rivest, R.L.: Making Mix Nets Robust for Electronic Voting by Randomized Partial Checking. In: USENIX Security Symposium 2002, pp. 339–353 (2002)
30. Lee, B., Kim, K.: Receipt-free electronic voting scheme with a tamper-resistant randomizer. In: Lee, P.J., Lim, C.H. (eds.) ICISC 2002. LNCS, vol. 2587, pp. 389–406. Springer, Heidelberg (2003)
31. Kiayias, A., Yung, M.: The vector-ballot e-voting approach. In: Juels, A. (ed.) FC 2004. LNCS, vol. 3110, pp. 72–89. Springer, Heidelberg (2004)

32. Kiayias, A., Korman, M., Walluck, D.: An Internet Voting System Supporting User Privacy. In: ACSAC 2006, pp. 165–174 (2006)
33. Andrew Neff, C.: A verifiable secret shuffle and its application to e-voting. In: ACM Conference on Computer and Communications Security 2001, pp. 116–125 (2001)
34. Okamoto, T.: Receipt-Free Electronic Voting Schemes for Large Scale Elections. In: Workshop on Security Protocols (1997)
35. Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Eurocrypt 1999 (1999)
36. Park, C.-s., Itoh, K., Kurosawa, K.: Efficient anonymous channel and all/Nothing election scheme. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 248–259. Springer, Heidelberg (1994)
37. Pedersen, T.P.: A threshold cryptosystem without a trusted party. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 522–526. Springer, Heidelberg (1991)
38. Pedersen, T.P.: Distributed Provers and Verifiable Secret Sharing Based on the Discrete Logarithm Problem, PhD Thesis, Aarhus University (1992)
39. Rabin, M.: Transactions protected by beacons. *Journal of Computer and System Sciences* 27, 256–267 (1983)
40. Sako, K., Kilian, J.: Secure voting using partially compatible homomorphisms. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 411–424. Springer, Heidelberg (1994)
41. Sako, K., Kilian, J.: Receipt-Free Mix-Type Voting Scheme - A Practical Solution to the Implementation of a Voting Booth. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 393–403. Springer, Heidelberg (1995)
42. Schoenmakers, B.: A simple publicly verifiable secret sharing scheme and its application to electronic voting. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, p. 148. Springer, Heidelberg (1999)

On Optical Mark-Sense Scanning

Douglas W. Jones*

University of Iowa, Iowa City IA 52242, USA

Abstract. Optical mark-sense scanning has led to a resurgence in the use of paper ballots in the United States, despite a century of strong competition from paperless direct-recording voting systems. By the time mark-sense technology emerged, procedural measures had been developed to counter most of the vulnerabilities of paper ballots. Automatic counting of paper ballots poses technical and legal problems, but by counting the paper ballots automatically in the presence of the voter, mark-sense systems address some of the remaining problems with paper ballots. The best current technology uses precinct-count optical scanners to capture pixelized images of each ballot and then process the marks on that image. While this technology may be among the best voting technologies available today for the conduct of complex general elections, it faces one significant problem, access to voters with disabilities. There are promising solutions to this problem, but work remains to be done.

1 Paper Ballots

Considerable effort has gone into developing paperless voting systems over the past century, but paper ballots have proven to be a remarkably durable voting technology. Mechanical voting machines, first used in the 1890s [16], promised to eliminate the paper ballot. By the 1960's, when Joseph Harris introduced the Votomatic punched-card voting system [30], mechanical voting machines had displaced paper throughout most of the United States.

Several mark-sense scanning systems were introduced at about the same time that could directly read and tabulate marks made on paper ballots. Between these systems, paper ballots made a strong comeback in the last three decades of the 20th century. By 1988, machine-counted paper ballots were being used by almost half of the electorate in the United States, while mechanical voting machines were used by about one third [41].

The second technology to challenge paper ballots was the direct-recording electronic voting machine. While there is one 19th century antecedent for this technology [45], the first successful application of this idea was the Video-Voter, patented in 1974 [37]. By 1988, this new technology had only very limited market penetration, but by 2004, almost one third of the electorate in the

* This material is based upon work supported by the National Science Foundation under Grant No. CNS-052431 (ACCURATE). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

United States was using direct-recording electronic voting machines, while an equal number were using optical mark-sense machinery [23]. By this point in time, hand-counting, mechanical voting machines and punched cards were all in retreat.

2 The Decline and Re-emergence of Paper

The decline in hand-counted paper ballot use in the United States followed from two distinct causes. The first is the perception that was very widespread a century ago that mechanized vote tabulation was inherently more resistant to fraud than hand-counted paper ballots. This position is very evident in several important reports from the 1920s and 1930s, such as [46] and [29] (see pages 370 to 375).

The second reason for the decline of hand-counting is the complexity of general elections in the United States. Where much of the world puts only one contest on the ballot, general election ballots in the United States frequently contain many referenda as well as partisan races for offices from the national level down to the most local, and as many as ten parties compete in many races.

It is straightforward to hand count ballots with only one race on the ballot and only a few candidates. A typical methodology is to sort the ballots into piles by candidate and then count the pieces of paper in each pile. Such a count is fairly easy to observe and check. In contrast, there is no simple method to quickly and accurately hand count complex general election ballots.

A key to the survival of paper ballots was the development of the Australian system of secret balloting. In this system, ballots listing all qualified candidates are printed at government expense. Ballots are then distributed to voters at polling places, where voters mark their ballots in the privacy of voting booths.

The Australian state of Victoria adopted this idea in 1856 [3], but its spread outside of Australia was slow. The British adopted this system in 1872 [4]. By 1892, the same year that mechanical voting machines saw their first use in the United States, the Australian model was in use in over 80 percent of the United States [18].

While the Australian secret ballot requires no technology more advanced than the printing press, as suggested by Figure 1, it is a sophisticated invention. The sophistication is procedural, not technological. There are specific procedural countermeasures to each of many threats to the integrity of an Australian secret ballot election. Many of these defensive measures have been known for decades [29]. Typical threats and defensive measures are summarized in the following paragraphs:

Ballot box stuffing, that is, the addition of pre-voted ballots to the ballot box, usually by corrupt election officials. To defend against this, the number of pollbook signatures should be compared with the number of ballots in the ballot box at the close of the polls. Ideally, there should be incident reports explaining any discrepancies, such as voters who fled after signing the pollbook without voting. All of these records should be public, so that the existence of problems is exposed.

OFFICIAL BALLOT Random County, Somestate	
INSTRUCTIONS: To vote for some candidate, make an X in the box beside the name of that candidate.	
<hr/> <p>PRESIDENT (vote for one)</p> <p><input type="checkbox"/> G. Washington</p> <p><input type="checkbox"/> A. Lincoln</p> <p><input type="checkbox"/> _____ (write in)</p>	<p>U.S. CONGRESS (vote for one)</p> <p><input type="checkbox"/> S. Rayburn</p> <p><input type="checkbox"/> J.G. Cannon</p> <p><input type="checkbox"/> N. Longworth</p> <p><input type="checkbox"/> _____ (write in)</p>

Fig. 1. An Australian secret ballot, that is, a ballot with the names of all qualified candidates printed on it. This example is for a fictional and greatly simplified general election with two different races on the ballot.

Ballot box substitution and Pollbook alteration allow the above check to be defeated. To prevent this, all processes should be open to public observation and where this is difficult, all materials should be in the joint custody of mutually distrustful adversaries such as members of opposing parties. Complete records of the chain of custody need to be maintained for all critical materials, and these should be public.

Ballot alteration during the count has been reported in some elections. No pens, pencils or erasers should be allowed within reach of the tellers who handle ballots, and tellers should wear white gloves or accept manicures from adversaries. This latter measure prevents hiding bits of pencil lead under fingernails.

Clerical Errors can corrupt the count, and where small errors are common, election manipulation can be disguised as error. To prevent errors in the count, tellers should sort ballots by how they are marked and then count the number of ballots in each pile. This procedure is comparable to the way large quantities of money are usually counted. As with money, counting does not alter what is being counted, so in the event of any controversy about the count, the process can be repeated.

Biased Counting is possible. For example, tellers can strictly apply the law on proper ballot markings for ballots they disapprove of, while generously interpreting voter intent for ballots they like. To defend against this, tellers should work in pairs made of representatives of opposing parties. While sorting ballots, they should sort disputed ballots separately from ballots they agree on. Disputed ballots should be further segregated by the nature of the dispute. The official

record of the count should then include the number of ballots in each disputed category. The purpose of this is to expose the existence of bias and the frequency with which voters mark ballots in ways subject to dispute. In the event of any controversy, the entire count can be redone.

Chain Voting is the most sophisticated fraud technique that has been employed against Australian ballot elections; it has been well documented for well over half a century [29] (see pages pages 40, 298, 299 and 373). The organizer of the chain needs one valid ballot to begin with. He then marks this ballot and gives it to a voter willing to participate in the fraud. With each participant, the organizer instructs the participant to vote the pre-voted ballot and bring back a blank ballot from the polling place. Voters are paid for the blank ballot. The best defense against chain voting involves printing a unique serial number on a removable stub on each ballot. When ballots are issued to voters, the stub numbers should be recorded. No ballot should be accepted for deposit in the ballot box unless its stub number matches a recently issued number. Finally, to preserve the voter's right to a secret ballot, the stub should be torn from the ballot before it is inserted in the ballot box.

Punched card ballots and optical mark-sense ballots are simple variations on the Australian secret ballot. All of the procedural defenses of the Australian method apply to these new technologies, and automated ballot counting addresses the single most challenging feature of the Australian model when applied to an American general election, the presence of many tens of different races on each ballot. Placement of multiple races on one ballot makes manual counting both error prone and time consuming. It is so time consuming that observers rarely stay through the entire process.

3 Mark-Sense Ballots

Practical mark-sense scanners were first developed for educational testing, but by 1953, proposals for mark-sense ballots were being advanced [36]. The Norden Electronic Vote Tallying System was the first system to apply optical mark sensing to ballots [44] [20] (see pages 25-27 and 55). Both of these early systems required the use of special inks, but the Votronic, patented in 1965, sensed ordinary pencil marks [31] [38] (see page 27).

On these ballots and their successors, preprinted *voting targets*, indicate where the voter is to mark the ballot. Scanners developed between the 1960s and the 1990s required that all voting targets be aligned in vertical tracks, one per sensor assembly on the scanner. One or more additional tracks of index marks define the positions of the voting targets along each track. Finally, it is common to include an index mark at the top and bottom of each track in order to test for faulty sensors and check for any skew in feeding the ballot through the scanner. Figure 2 illustrates such a ballot layout.

While most mark-sense ballots use oval or elliptical targets, with the index marks on opposite edges of the ballot, there are alternatives. The Optech line of scanners, for example, use a broken-arrow as a target, instructing voters to

Fig. 2. A mark-sense version of the ballot from Figure 1. Index marks have been added around the edges to allow the scanner to locate the voting targets, the form of the targets has been changed, and the instructions have been changed to match the requirements of the scanner.

connect the two halves of the arrow in order to cast a vote. The two halves of the arrow in this system are used as index marks to locate the target area between them [39].

It is important to note that the *sensitive area* of the ballot, where marks will be sensed as votes, need not be the same as the area outlined by the voting target. Rather, the sensitive area is defined by the geometry of the sensor itself and the positions of the index marks. The ballot shown in Figure 2, for example, has index marks defining 8 rows and 6 columns, for a total of 48 sensitive areas. Of these, only 7 have an assigned meaning on this ballot.

4 What Is a Vote?

The instructions for marking a ballot prescribe some mark, for example, filling in the oval voting target or connecting the two halves of a broken arrow voting target. This *prescribed mark* is designed to be reliably counted by the sensor system and easily explained to voters. When the voting target is an oval or ellipse, as shown in Figure 2, the prescribed mark is generally a perfectly filled oval, as shown in Figure 3a.

The universe of all possible markings of a particular sensitive area on a ballot can be classified as either *legal votes* if the law accepts them as indicating votes

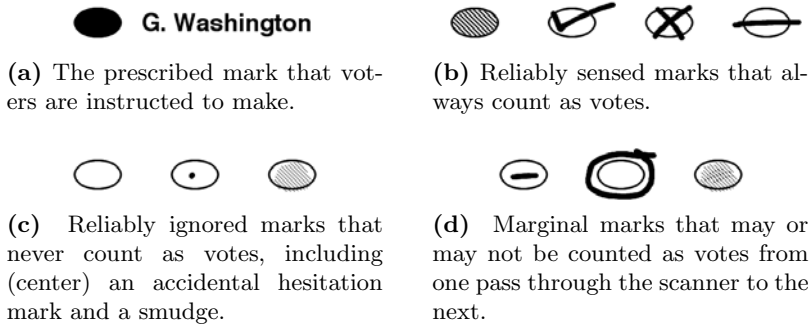


Fig. 3. Classes of ballot markings, distinguished by how they are recognized by a typical optical mark-sense ballot scanner. Illustration based on results for the Election Systems and Software model 650 scanner, as reported in [33].

and *legally ignored* if the law considers them not to be votes. In addition, independently of whether the mark is or is not considered a vote by the law, it may be classified according to how the scanner interprets it.

Marks may be *reliably sensed*, if every time that mark is seen by a properly adjusted scanner, it is always counted as a vote. In general, the scanners the author has tested reliably sense a variety of marks. The results in Figure 3b are typical. In general, attempts to duplicate the prescribed mark using pencil are reliably sensed, regardless of what kind of marker is prescribed. In addition, checks, X marks and single pen strokes made with the marker originally prescribed by the developer of the voting system are reliably sensed. Some scanners exhibit considerable variation in sensitivity [34] (see Exhibits 5 to 7).

A mark is *reliably ignored* if it is never seen by the scanner. Of course, an unmarked voting target should be reliably ignored, but so should flecks in the paper, smudges and *hesitation marks*, as suggested in Figure 3c. Hesitation marks are a fairly common artifact found on mark-sense forms. They are the result of people using the marker as a pointer to point to targets as they consider whether to mark those targets.

Finally, there are invariably some *marginal marks*. These are marks that may or may not be sensed, depending on when they are run through the voting machine and which particular sensor the mark happens to be seen by. Dark smudges, short lines within the voting target and marks entirely outside but close to the voting target are frequently marginal. Attempts by voters to imitate the prescribed mark using red ink are particularly problematic on scanners that use red or infrared light to illuminate the page.

Obviously, the prescribed mark should be both a legal vote and reliably sensed, as should all approximations of the prescribed mark likely to be made by voters. Similarly, smudges and similar accidental markings should be both legally ignored and reliably ignored.

There is a three-way interaction here between the voting system, the law and the voters. Unfortunately, there is no consensus about how the set of markings

on a ballot should be legally classified. The most dangerous approach is known as the *machine model*. This defines as legal votes whatever the machine accepts [14]. The machine model does not allow for the existence of marginal marks, nor does it provide any criteria for judging whether the scanners conform to the law.

At the other extreme are laws that enumerate the types of markings that are legal votes. Consider, for example, Michigan’s rules as of 2004 [9]. These rules, and the law on which they are based, do not distinguish between the sensitive area and the voting target. They declare some markings to be legal votes that a scanner may miss, while declaring other marks to be legally ignored even though a scanner might count them, as illustrated in Figure 4.

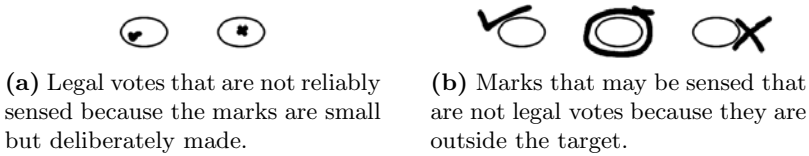


Fig. 4. Problematic ballot markings under Michigan law if scanned on the Election Systems and Software model 650 scanner. Illustration combines results from [33] with the law in [9].

These discrepancies between legal votes and what the scanner counts are troublesome, but they are not necessarily a major problem. So long as real voters rarely make these marks, they can be ignored except in very close races, and when there is such a race, hand recounts can be used to resolve them. It is, however, important to know what fraction of ballot markings are problematic. Without knowing this, we cannot evaluate human factors problems with the ballots, nor can we determine when to call for a hand recount.

5 Scanning Technology

The first generation of mark-sense scanners employed a single sensing element per vertical track down the ballot, as illustrated in Figure 5. The Votronic system, employing this sensor, saw widespread use from 1964 into the 1970s [20] (see pages 27 56, 59-60 and 62). While the Votronic sensor employed an ordinary light bulb to illuminate the ballot, the silicon photosensor had its peak sensitivity in the infrared.

Infrared photosensors remain in use to this day. The advantage of such sensors is that they allow voting targets to be printed red ink or some other ink invisible to the photosensor, thus simplifying sensor calibration. The disadvantage of this is that voters are ill equipped to judge whether the marks they have made are reflective at infrared wavelengths.

The disadvantages of infrared scanning are clearly documented in the Optech 4C patent, where it is noted that “most common pens do not use ink that adsorbs infra-red light” [43]. This is particularly troublesome with postal ballots

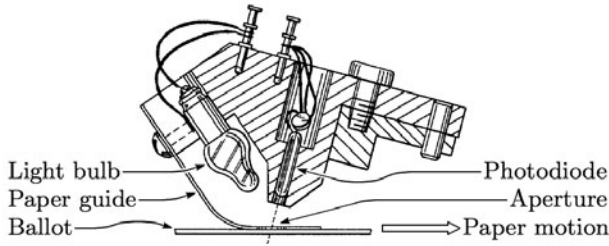


Fig. 5. Section through the Votronic optical mark-sensing assembly along the direction of paper motion, based on [31]

because it is difficult to control what kind of ballot markers are used outside the controlled context of the voting booth.

While it is easy to imagine the sensitive area of the ballot having sharp edges, most scanners using discrete sensors have relatively broad scanning tracks and are more sensitive toward the center of the track than the edges. This is a natural result of scanning through a circular aperture or an aperture with circular corners, as illustrated in Figure 5. When the scanner does not physically contact the ballot, for example, to avoid smudging any marks that might be present, the edge of the sensitive area is not sharply defined.

In the direction along the scanning track, the boundary of the sensitive area is defined by the temporal response of the scanning circuitry and by how the analog signal from the sensor is sampled. In order to avoid sensing smudges, for example, scanners can be designed to check not only the intensity signal but the derivative of that signal, so that a faint mark with sharp edges is counted even while a darker smudge is ignored.

Any systems that use paper must account for the fact that paper is not dimensionally stable. Paper expands with increasing humidity, with dimensional changes approaching one percent [27]. The placement of index marks along the long dimension of the page allows the scanner to automatically compensate for changes in that dimension, and the use of voting targets and scanning tracks that are wide along the short dimension of the page allows dimensional changes along that dimension to be largely ignored.

The development of mass produced fax machines and page scanners allowed more than one sensing element to be positioned over each track of the ballot. With this change, mark-sensing shifts from the domain of signal processing to the domain of image processing. The American Information Systems Model 100 Precinct Ballot Scanner was one of the first to employ imaging technology [1]. Although this fact and the pixel-counting threshold algorithm it used to distinguish between different types of ballot markings can be inferred from manuals dating to 1997, public disclosure of these algorithms only occurred in the patent issued in 2005 [22].

The emphasis in the design of the AIS Model 100 was on scanning ballot formats originally developed for discrete-sensor scanners. These ballots included a complete suite of index marks, with additional marks allowing the scanner to


Election 012 Precinct 345 Style 678	 OFFICIAL BALLOT Random County, Somestate
<hr/> INSTRUCTIONS: To vote for some candidate, fill in the oval by that candidate's name.	
PRESIDENT (vote for one)	U.S. CONGRESS (vote for one)
<input type="radio"/> G. Washington	<input type="radio"/> S. Rayburn
<input type="radio"/> A. Lincoln	<input type="radio"/> J.G. Cannon
<input type="radio"/> _____ (write in)	<input type="radio"/> N. Longworth
<input type="radio"/> _____ (write in)	<input type="radio"/> _____ (write in)

Fig. 6. A ballot based on those in Figures 1 and 2 designed for scanning using imaging technology. Two fiducial marks in opposite corners allow the analysis software to interpolate the voting target locations. A bar code allows the software to distinguish between different ballot styles that might be used. The form shown here is based loosely on that used by the Fidlar and Chambers AbScan-T [5].

detect the ballot orientation. In addition, one scanning track is used as what is essentially a long low-density bar code to encode precinct-number and ballot style. Another voting system vendor, Fidlar and Chambers, was not constrained by compatibility with the past.

The Fidlar and Chambers AbScan-T absentee ballot system came to market in early 2000 [13]. This system used a commercial off-the-shelf flatbed scanner and page feeder to scan ballots for processing on a personal computer [7]. The ballot layout used on this machine is illustrated, in reduced scale, in Figure 6.

The Fidlar ballot included two obvious changes from earlier mark-sense ballots. First, it incorporated a conventional looking bar code where earlier scanners had used code tracks or code regions that a naive observer might not recognize as a bar code. Second, instead of index marks around the edges of the ballot, it had just two *fiducial marks* on opposite corners of the ballot. The term *fiducial mark* comes from the field of photogrammetry; it refers to a mark used as a point of reference for locating or measuring the locations of features in an image. Having located these marks, the ballot analysis software can use them to identify the ballot scale and orientation before searching for the bar code and voting targets.

Scanners such as the AIS Model 100 rely on technology originally developed for fax machines to see the ballot in black-and-white, with no shades of grey. At most, such scanners allow the overall black-white threshold to be set once, before the capture of a ballot image for processing. Image scanners offer the potential

to dynamically adapt to the background color of the page and judge the presence of markings on the ballot on the basis of criteria more sophisticated than merely counting the black pixels within some predefined sensitive area around each voting target.

To date, commercial mark-sense ballot scanners generally recognize marks by counting the number of dark pixels in the sensitive area enclosing each voting target. This approach reappears, for example, in the claims of a patent issued in 2006 [24]. An experimental ballot tabulator has been demonstrated that uses a significantly more sophisticated mark-recognition algorithm. It first analyzes the image of the ballot using an edge detection algorithm, and then locates closed rings of edges in the image. Unmarked voting targets appear as pairs of concentric rings, while completely filled targets appear as single rings. With this algorithm, the targets themselves can serve as fiducial marks, allowing easy recovery of data from ballots images even if they are significantly distorted. Reliance on edge detection eliminates sensitivity to the background illumination level. The initial demonstration of this algorithm used a video camera to read the ballots from a distance using ambient lighting and off-axis viewing [19].

There is clearly room for considerable elaboration on the use of edge detection in ballot image analysis. Consider the problem of dealing with non-standard but legal marks such as were illustrated in Figure 3b; a simple search for closed rings in the image will not count these as votes.

6 Second Chance Voting

With the advent of microprocessors, scanners became inexpensive enough that it was practical to install one ballot tabulator in each precinct, integrated into the ballot box. The Gyrex MTB-1, introduced in the mid 1970s, was an important early example of such a machine [40] [38] (see page 42). This used a very primitive bar code to encode ballot style, thus allowing the use of multiple ballot styles in one precinct, as is required for partisan primary elections, and it incorporated a printer so that it could print results immediately when the polls were closed. In combination, these features address an important category of threat:

Ballot alteration or ballot substitution: Immediate scanning eliminates the opportunity to alter ballots or substitute alternate ballots between the time they are voted and the time they are counted. In effect, from the moment the ballots are scanned onward, there are two independent records of the vote, one on the marked paper ballot itself, and one in the scanner's memory. Of course, problems will be detected only if the paper ballots in the ballot box are actually examined. In 1965, California enacted legislation requiring such an examination in randomly selected precincts after every election; more recently, several other states have enacted such legislation [8].

A second feature of precinct-count equipment emerged later, the ability to return ballots to the voter. This emerged with the CESI Optech I scanner [28] [20] (see pages 67-68). While this may have originated as a way to clear jams and handle misread ballots, it quickly emerged that one of the most valuable

features of precinct-count scanners was that they could reject overvoted ballots, returning them to the voter. Direct recording mechanical and electronic voting machines have routinely offered this protection since the 19th century [42]. With the passage of the Help America Vote Act of 2002 (HAVA), all voting equipment used at polling places in the United States is required to offer this protection [6].

Most precinct-count scanners in current use can also return ballots that scan as blank. This offers protection for voters who use ballot markers invisible to the scanner, and it offers protection for those who completely misunderstand the ballot marking instructions by marking entirely outside the sensitive areas on the ballot. Generally, when ballots contain multiple races, as in general elections in the United States, it has not proven to be useful to return ballots where votes are found in some races but not in others. In such elections, most voters abstain some races.

When ballots are centrally counted, for example, where postal voting is used, voters have no equivalent protection against overvoting. HAVA suggests that voter education and instructions can substitute for this, but there is ample evidence that this is not true. The best current practice for postal voting is to require that the ballot scanner sort out all ballots that scan as blank or contain overvotes. These ballots should then be examined by the canvassing board to determine if they contain indications of voter intent. Typically, in general elections in the United States, the canvassing board must examine around 4 percent of the ballots [21].

Imaging scanners allow an alternative approach to resolving questionable markings on centrally counted ballots. Instead of sorting out the ballots requiring human inspection, the vote tabulation system can present scanned images of these ballots to the canvassing board for resolution. This was done in the 2007 Scottish Parliamentary elections [10], and the same functionality is present in the Ballot Now system from Hart Intercivic [2]. When the actual ballot is not examined to resolve the markings on the ballot, it is within reason to ask that any audit of the election inspect the authenticity of the ballot images that were examined as well as how each problematic marking was resolved.

7 Human Factors

All voting systems, from the most primitive to the most technological, are data-capture systems. This is true whether we ask voters to enter their selection directly into computers, to mark their selections on paper for manual processing, or to mark their selections on paper for scanning by a vote tabulating machine.

The single greatest strength of mark-sense voting is that the basic medium, pen or pencil marks on paper, is one with which the vast majority of voters are expert. Most people began their formal training in the use of this medium in kindergarten, and it is fair to say that the average person has far more training and experience with making and interpreting marks on paper than with any other data recording medium.

Despite this familiarity, there is ample evidence that very small changes in voter instructions can lead to significant changes in the likelihood that voters

will correctly express their intent [33]. For example, the instructions on the ballot used in Maricopa County, Arizona, on September 7, 2004 said “TO VOTE: Complete the arrow(s) pointing to your choice with a single line,” with appropriate illustrations. When the ballot tabulating system was tested, it was found that a single line made with a common ballpoint pen was a marginal mark [34] (see Exhibit 8). Fortunately, most voters scrupulously darken their marks, but the instruction to make a single line may have mislead an unknown number of voters.

The author suggests that a voting target be printed in the pollbook next to the name of each eligible voter. On signing the pollbook, the voter could then be asked to properly mark this oval. Doing this would give pollworkers an opportunity to observe any difficulty people have making an appropriate mark in a context where there is no threat to the voter’s right to a secret ballot.

The 1990 and 2002 federal voting system standards required that mark-sense scanners distinguish between the prescribed mark, on the one hand, and smudges or creases, on the other. They did not, however, require any exploration of the universe of other markings voters might make [12] [17]. Unfortunately, while the 2005 guidelines incorporated an extensive human-factors section, this is focused largely on handicapped accessibility and it does not alter the requirements for mark-sense tabulation accuracy [15].

It is noteworthy that post election auditing of mark-sense ballot tabulation systems can do more than audit the correctness of the count. A properly conducted post-election audit of mark-sense ballots should also note the number of ballots marked with nonstandard markings. In contrast, the audits proposed for direct-recording electronic voting systems tend to focus exclusively on the count and exclude human-factors issues [32].

8 Disability Problems

Mark-sense ballots do pose some difficult challenges. In jurisdictions requiring multilingual ballots, adding an extra language to each ballot adds clutter, and this decreases readability. This is tolerable with bilingual ballots, but where three or more languages are required, readability declines rapidly.

A more serious problem involves access for voters with disabilities, particularly blind voters, but also those with motor disabilities and poor eyesight. Voters who need large print can be aided by providing them with magnifying glasses. While these are somewhat cumbersome, they are also a very familiar technology. Similarly, voters with motor disabilities can be provided with transparent ballot overlays to protect the ballot from stray marks and scribbles. These measures meet the needs of the majority of voters who might otherwise need assistance in casting their votes, but they are rarely provided in modern polling places.

Since the passage of the Help America Vote Act of 2002 in the United States, the problem of providing access to the blind has frequently solved by providing one handicapped accessible voting system per polling place. In many jurisdictions, a direct-recording electronic voting machine is used for this purpose. Use

of multiple vote recording systems threatens voter privacy, particularly if only a few voters use the accessible system while the majority use mark-sense ballots.

Voter privacy is improved if the ballots voted by blind voters or others needing assistance are merged with all other ballots voted at the same location prior to tabulation. This idea has led to the development of several accessible ballot marking devices that allow blind voters to mark paper ballots. The AutoMark [25] and the Vote-PAD [11] are the two most widely discussed. These devices seek to achieve the same goal, but they do so in radically different ways.

The AutoMark uses any of several input devices to capture the voter's choice, and then it uses ink-jet printer technology to record that choice on a standard mark-sense ballot. The input devices are typical of direct-recording electronic voting machines, enough so that this machine could be classified as an *indirect-recording* electronic voting machine.

In contrast, the Vote-PAD is a *tactile ballot* [26]. That is, it is a template that fits over the ballot, allowing the voter to mark the ballot through holes in the template. This alone is sufficient to aid most sighted voters with motor disabilities. For those who cannot read the ballot, a recorded script is provided to narrate the ballot.

Tactile ballots have been used with great success in many countries, but in most of these cases, elections typically involve a single race with only a few candidates. When used for a general election in the United States, the audio narration of the ballot can easily take 15 minutes, and audio instruction for how to navigate a large ballot is both cumbersome and error prone.

There is clearly room to explore other solutions to making mark-sense ballots more accessible. One proposal combines the mechanics of a tactile ballot with the mechanism of a graphics tablet. With this system, instead of following an audio script, the voter is free to explore the ballot by moving a sensing wand over the template. The system senses the location of the wand and reads whatever ballot position the voter selects, reporting on whether or not it has already been voted. While such a mechanism might cost considerably more than a tactile ballot, it would be considerably less expensive than an AutoMark machine [35].

9 Conclusion

Optical mark-sense vote tabulation will remain widely used into the indefinite future. It is the only technology for automatic vote tabulation that is applicable to postal voting, and there are promising technologies available to permit disabled voters to use mark-sense ballots at polling places. From a security perspective, it can be judged highly secure if appropriate post election audit procedures are used.

As such, optical mark-sense systems are one of the best voting technologies available for complex general elections. There are, however, several areas where additional work needs to be done. First among these is the development of assistive technologies that are both inexpensive and effective for complex general elections.

A second frontier lies at the heart of the mark-sensing mechanism itself. Mark-sensing algorithms that make intelligent decisions based on image analysis are currently in their infancy. Only the most tentative experiments have been made with applying image processing techniques to this area.

As with all voting technologies, there are major problems with the diffusion of best practices. Some jurisdictions have long employed sound practices for post-election audits, processing of overvoted ballots, and wording of ballot instructions, while these same practices remain essentially unknown in other jurisdictions. Voting system standards, state oversight and further professionalization of voting system administrators will all play a role in solving this problem.

References

1. AIS Model 100 Precinct Ballot Scanner Operator's Manual, Version Release 2.1, American Information Systems (August 1997)
2. Ballot Now Intelligent Ballot Management, Hart InterCivic (2005)
3. The Electoral Act of 1856, Victoria, Australia (1856)
4. Procedures at a General Election, British Department for Constitutional Affairs, August 17, 2001. Paragraphs 11.14 through 11.18 (2001)
5. Official Ballot General Election, Clay County Iowa, November 7 (2000)
6. The Help America Vote Act of 2002, Public Law 107-252, 107th Congress, Title III, Sec 301(a)1(A) (2002)
7. Minutes of Examination and Test, Board of Examiners for Voting Machines and Electronic Voting Systems, Iowa Secretary of State, October 17 (2000)
8. Manual Audit Requirements, Verified Voting Foundation, November 1 (2006)
9. Determining the Validity of Optical Scan Ballot Markings, State of Michigan Department of State, July 18 (2006)
10. Managing the Scottish Parliamentary and local government elections guidance for Returning Officers, UK Electoral Commission, Section F, 3, pp. 181–186 (2007)
11. Vote-PAD (Voting-on-Paper Assistive Device) - Independent Voting for People with Disabilities, Vote-PAD Inc. (2007)
12. Performance and Test Standards for Punchcard, Marksense, and Direct Recording Electronic Voting Systems, Federal Election Commission, Section 3.2.5.2.1 (January 1990)
13. Contract between Fidler & Chambers Co. and the Surry County Board of Commissioners, Surry County, North Carolina, January 26 (2000)
14. U.S. Court of Appeals, 11th Circuit, December 6, 2000, Touchston and Shepperd vs. Michael McDermott, No. 00-15985 (2000)
15. Voluntary Voting System Guidelines, U.S. Election Assistance Commission (2005)
16. Voting Machines, Encyclopaedia Britannica, 11th edn., vol. 28 (1910)
17. Voting System Performance and Test Standards, Federal Election Commission, vol. I, Section 3.2.4.2.3 (2002)
18. Ackerman, S.J.: The Vote that Failed, Smithsonian (November 1998)
19. Adi, A., Adi, W.: Demonstration of Open Counting: A Paper-Assisted Voting System with Public OMR-At-A-Distance Counting. In: VoComp 2007, Portland, Oregon, July 17 (2007)

20. Arnold, E.G.: History of Voting Systems in California, California Secretary of State Bill Jones (June 1999)
21. Berghoefer, F.: Transcript of oral testimony, Technical Guidelines Development Committee Public Data Gathering Hearings, September 20 (2004)
22. Bolton, S., Cordes, T., Deutsch, H.: Method of Analyzing Marks Made on a Response Sheet, U.S., February 15 (2005) Patent 6,854,644
23. Brace, K.W.: Overview of Voting Equipment Usage in United States, Direct Recording Electronic (DRE) Voting, statement to the United States Election Assistance Commission, May 5 (2004)
24. Chung, K.K., Dong, V.J., Shi, X.: Electronic Voting Method for Optically Scanned Ballot, U.S., July 18 (2006) Patent 7,077,313
25. Cummings, E.M.: Ballot Marking System and Apparatus, U.S., July 25 (2006) Patent 7,080,779
26. Davidson, A., Ricks, S.: Tactile Ballot for the Visually Impaired of Marion County, Oregon, Appendix C of Independent, Secret and Verifiable - A guide to Making Voting an Independent and Accessible Process for People who are Blind and Visually Impaired, American Council for the Blind (September 2002)
27. Dwan, A.: Paper Complexity and the Interpretation of Conservation Research. *Journal of the American Institute for Conservation* 26(1), 1–17 (1987)
28. Charles Fogg, M., Krieger, C.F., Veale, J.R.: System and Method for Reading Marks on a Document, U.S., October 23 (1984) Patent 4,479,194
29. Harris, J.P.: Election Administration in the United States. The Brookings Institution (1934)
30. Harris, J.P.: Data Registering Device, U.S., August 17 (1965) Patents 3,201,038, March 15 (1966) 3,240,409
31. Holzer, G., Walker, N., Wilcock, H.: Vote Tallying Machine, U.S., November 16 (1965) Patent 3,218,439
32. Jones, D.W.: Auditing Elections. *Comm. ACM* 47(10), 46–50 (2004)
33. Jones, D.W.: Observations and Recommendations on Pre-election Testing in Miami-Dade County, September 9 (2004)
34. Jones, D.W.: Regarding the Optical Mark-Sense Vote Tabulators in Maricopa County, statement submitted to the Arizona Senate Government and Accountability Committee, January 12 (2006)
35. Jones, D.W.: System for Handicapped Access to Voting Ballots, U.S., November 14 (2006) Patent 7,134,579
36. Keith, H.R.: Electrical Vote Counting Machine, U.S., June 12 (1956) Patent 2,750,108
37. McKay, R.H., Ziebold, P.G., Kirby, J.D., Hetzel, D.R., Snyder, J.U.: Electronic Voting Machine, U.S., February 19 (1974) Patent 3,793,505
38. Moloney, M.A.: Mechanized Vote Recording: A Survey, Research Report No. 116, Legislative Research Commission, Frankfort Kentucky (May 1975)
39. Narey, J.O.: Ballot for use in Automatic Tallying Apparatus and Method for Producing Ballot, U.S., March 21 (1989) Patent 4,813,708
40. Narey, J.O., Saylor, W.H.: Ballot Tallying System Including a Digital Programmable Read-Only Control Memory, a Digital Ballot Image Memory and a Digital Totals Memory, U.S. Patent 4,021,780
41. Saltman, R.G.: Accuracy, Integrity, and Security in Computerized Vote-Tallying, NBS Special Publication 500-158, National Bureau of Standards (August 1988)

42. Spratt, H.W.: Improvement in Voting Apparatus, U.S., January 12 (1875) Patent 158,562
43. Stewart, J.D.: Device for Optically Reading Marked Ballots using Infrared and Red Emitters, U.S., September 28 (1993) Patent 5,248,872
44. Weik, M.H.: A Third Survey of Domestic Electronic Digital Computing Systems, Ballistic Research Laboratories Report No. 1115, pp. 719–723 (March 1961)
45. Wood, F.S.: Electric Voting-Machine, U.S., December 20 (1898) Patent 616,174
46. David Zuckerman, T.: The Voting Machine. Political Research Bureau of the Republican County Committee of New York (January 1925)

On Some Incompatible Properties of Voting Schemes

Benoît Chevallier-Mames¹, Pierre-Alain Fouque², David Pointcheval²,
Julien Stern³, and Jacques Traoré⁴

¹ DCSSI

`Benoit.Chevallier-Mames@sgdn.pm.gouv.fr`

² ENS – CNRS – INRIA

`Pierre-Alain.Fouque,David.Pointcheval@ens.fr`

³ Cryptolog International

`Julien.Stern@cryptolog.com`

⁴ Orange Labs – France Telecom R&D

`Jacques.Traore@orange-ftgroup.com`

Abstract. In this paper, we study the problem of simultaneously achieving several security properties, for voting schemes, without non-standard assumptions. More specifically, we focus on the universal verifiability of the computation of the tally, on the unconditional privacy/anonymity of the votes, and on the receipt-freeness properties, for the most classical election processes. Under usual assumptions and efficiency requirements, we show that a voting system that wants to publish the final list of the voters who actually voted, and to compute the number of times each candidate has been chosen, we cannot achieve:

- universal verifiability of the tally (UV) and unconditional privacy of the votes (UP) simultaneously, unless *all* the registered voters actually vote;
- universal verifiability of the tally (UV) and receipt-freeness (RF), unless private channels are available between the voters and/or the voting authorities.

1 Introduction

1.1 Motivations

A huge number of properties for voting schemes have been proposed so far: and namely, *universal verifiability* (UV), the *unconditional privacy/anonymity* of the votes (UP), receipt-freeness (RF), and incoercibility.

Some properties seem quite important because usual systems and/or paper-based systems achieve them, and some other seem more theoretical because they are not (efficiently) satisfied in existing schemes: people expect much more from electronic voting schemes than from paper-based systems: the best example is the universal verifiability, which is definitely not satisfied with the paper-based voting systems, since one can supervise one place only. On the other hand, an

attack on an internet-based vote could be at a very large scale and thus much more damaging.

Furthermore, some properties are easily satisfied by using physical assumptions such as voting booths, while they are difficult if one can vote from home: this is the case of incoercibility. Since cryptography is usually very powerful and makes possible some paradoxical things, one is tempted to build a system that achieves as many properties as possible, with as few assumptions as possible. But what is actually achievable?

1.2 Contributions

In this paper, we address this question: can we build a voting system that simultaneously satisfies several properties, without non-standard assumptions (such as physical assumptions)? More precisely, we focus on the large class of election systems that simply consist in counting the number of times that each candidate has been chosen (whatever the constraints on the choices may be) and want to be able to compute the list of the voters who actually voted. Such election rules are used in many countries (such as in France). On the one hand we study the universal verifiability (UV) and the unconditional privacy of the votes (UP), which is sometimes replaced by the unconditional anonymity of the voters. On the other hand, we consider the universal verifiability (UV) and the receipt-freeness (RF). In both cases, we show that we cannot simultaneously achieve the two properties without strong extra assumptions, such as secure channel between the voters and/or the authorities, which is unrealistic for efficient and practical protocols.

The universal verifiability and the unconditional privacy can actually be simultaneously satisfied if *all* the registered voters *do* vote; similarly the universal verifiability and the receipt-freeness can be simultaneously achieved if the voting transcript of a voter *does not* depend on the voter's vote, his secret, some personal possible private/random value, and additional public data only. It is well-known that using multi-party computation techniques a strongly secure voting scheme can be built, that achieves all the above ideal properties, but using secure channels between the parties (the voters and/or the authorities): efficient voting schemes that guarantee receipt-freeness or incoercibility [2,4,13,17,18,21] use such secure channels.

In the standard model we adopt below, we assume algorithmic assumptions only, but no secret channels nor physical assumptions such as tamper-resistant devices [18]. In addition, while studying the security properties of voting schemes, we try to explain why the traditional schemes, based on blind signatures, mix-nets or homomorphic encryption, satisfy these properties or not.

Having a clear view of which sets of properties are achievable has a practical significance: one can easily conceive that the properties required for a national election or for an internal company board vote are different. For instance, the unconditional privacy (UP) of the vote will be important (if not required) for national elections, while the receipt-freeness (RF) will not be as critical as it

may be difficult to buy votes on a very large scale without detection. For a board vote, a few number of voters typically have a very large number of shares, while the rest have a small number of shares. The major voters choices are often not private (let alone unconditionally private) because they can be inferred from the result of the vote. However, it may be tempting for a dishonest important voter, which could already have 40% of the shares, to buy the missing 10% to safeguard a majority. The receipt-freeness property is therefore more critical in that case.

1.3 Organization

The paper is organized as follows: first, in section 2, we give formal definitions to the above UV, UP and RF security notions. Then, we show the incompatibility results in section 3.

1.4 Notation

We use the following notation in the rest of the paper:

- L represents the list of the registered voters,
- V_i is a voter, who casts his ballot,
- \mathbf{V} is the list of the voters, who cast their ballots,
- v_i is the vote of voter V_i ,
- \mathbf{v} is the set of votes,
- r_i is the random coins of voter V_i ,
- \mathbf{r} is the set of the random coins,
- B_i is the transcript of V_i (that is the interactions between voter V_i and the voting authority, assumed to be public),
- \mathbf{B} is the set of transcripts, also known as the bulletin-board,
- T is the tally of the vote (the vector of the number of times that each candidate has been chosen),
- w, w' will denote the witnesses in some \mathcal{NP} - relations R and R' ,
- f, f', f'', g and h will be some functions.

Since we won't assume any private channel, any interaction can be assumed public, and also through the authority, and then included in the public transcript available on the bulletin-board. Furthermore, for practical reasons, the vote-and-go approach is often preferable, which excludes any complex interaction, but with the authorities only.

2 Security Notions

In this section, we formally define the most usual security notions: universal verifiability, unconditional privacy, and receipt-freeness.

2.1 Universal Verifiability of the Tally

This security notion tries to prevent dishonest voting authorities from cheating during the computation of the tally.

For example, voting schemes using blind-signature [8,16,20] cannot achieve this property since the authority can add some ballots and bias the tally. On the other hand, schemes using mix-nets [1,9,10,11,12,14,19,22] and/or homomorphic encryption [3,6,7] may provide it.

First, in order to universally check the validity and the correctness of a vote, one has to guarantee that a voter has not voted twice. Consequently, one needs to authenticate the transaction in some way. To this end, one needs to be able to verify both the link between the list of the registered voters L and the list of the transcripts \mathbf{B} (or the bulletin-board) in order to validate the vote, and the link between the bulletin-board and the computation of the tally T .

Definition 1 (Voting Scheme). *For a voting scheme to be practical and sound, it must hold the following properties.*

- Detection of individual fraud. *From a partial list of transcripts \mathbf{B} produced by $V_1, \dots, V_n \in L$, the voting authority should be able to determine whether a new transcript B produced by V_{n+1} is valid (well-formed and does not correspond to a double vote). More formally, there exists a boolean function f that can determine this fact,*

$$\left. \begin{array}{l} \forall n, \forall V_1, \dots, V_n, V_{n+1} \in L, \\ \forall \mathbf{B} \leftarrow V_1, \dots, V_n, B \leftarrow V_{n+1}, \\ f(\mathbf{B}, B) = \begin{cases} 0, & \text{if } V_{n+1} \in \{V_1, \dots, V_n\} \\ 1, & \text{if } V_{n+1} \notin \{V_1, \dots, V_n\} \end{cases} \end{array} \right\} \wedge B \text{ valid.}$$

We thus denote by \mathcal{L} the language of the bulletin-boards \mathbf{B} which are iteratively valid.

- Computation of the tally. *From the transcripts, the voting authority should be able to compute the tally, that is a vector of the number of selections for each candidates: there exists an efficient function f' that, from the bulletin-board \mathbf{B} , outputs the tally T ,*

$$\forall \mathbf{B} \in \mathcal{L}, f'(\mathbf{B}) = \sum_i v_i = T.$$

- Computation of the list of the voters. *From the transcripts, the voting authority should be able to determine the list of the voters who actually casted their ballots: there exists an efficient function f'' that, from the bulletin-board \mathbf{B} , extracts the sub-list \mathbf{V} of the voters,*

$$\forall \mathbf{B} \in \mathcal{L}, f''(\mathbf{B}) = \mathbf{V}.$$

When one wants the universal verifiability, everybody should be able to check the correctness/validity of the votes and of the computation of the tally and

the voters: the bulletin-board \mathbf{B} , the tally T and the list of the voters \mathbf{V} should rely in an \mathcal{NP} language \mathcal{L}' , defined by the relation R : there exists a witness w which allows an efficient verification. Furthermore, for any \mathbf{B} , the valid T and \mathbf{V} should be unique:

Definition 2 (Universal Verifiability (UV)). *Let R be the \mathcal{NP} -relation for the language \mathcal{L}' of the valid ballots and valid computation of the tally. A voting scheme achieves the universal verification property if only one value for the tally and the list of the voters can be accepted by the relation R , and the witness w can be easily computed from the bulletin-board \mathbf{B} using a function g :*

$$\begin{aligned} \forall \mathbf{B} \in \mathcal{L}, \exists! (T, \mathbf{V}) \text{ s.t. } \exists w \text{ s.t. } R(\mathbf{B}, T, \mathbf{V}, w) = 1 \\ \forall \mathbf{B} \notin \mathcal{L}, \forall (T, \mathbf{V}, w) \ R(\mathbf{B}, T, \mathbf{V}, w) = 0 \\ \forall \mathbf{B} \in \mathcal{L} \ R(\mathbf{B}, f'(\mathbf{B}), f''(\mathbf{B}), g(\mathbf{B})) = 1. \end{aligned}$$

Note that g is a function private to the authorities, to compute a short string (the witness) that allows everybody to check the overall validity, granted the public relation R .

The functions f , f' , f'' and g may be keyed according to the system parameters: g is clearly private to the voting authority, while f and f'' may be public (which is the case in schemes based on homomorphic encryption). The function f' is likely to be private.

2.2 Unconditional Privacy

First, one should note that this notion can not be achieved in a very strong sense: if all voters vote identically, the tally reveals the vote of each voter. Consequently, privacy means that nobody should learn more information than what is leaked by the tally. By unconditional privacy, we thus mean that nobody should be able to learn any additional information even several centuries after the voting process.

In voting schemes based on homomorphic encryption [3,6,7] privacy relies on computational assumptions, and is thus not unconditional. When mix-nets are used, this is the same, since the latter applies on asymmetric encryptions of the votes. On the other hand, voting schemes based on blind signatures can achieve this strong security notion, but under the assumption of anonymous channels, which are usually obtained with asymmetric encryption: unconditional privacy vanishes!

Definition 3 (Unconditional Privacy (UP)). *A voting scheme achieves the unconditional privacy if*

$$\mathcal{D}(\mathbf{v} \mid T, \mathbf{B}) \stackrel{p,s}{\equiv} \mathcal{D}(\mathbf{v} \mid T).$$

This equation means that the distribution of the votes, given the bulletin-board and the tally T is the same as without any additional information to the tally. The distance between these two distributions can be perfect or statistical, hence the s and p . But we of course exclude any computational distance.

2.3 Receipt-Freeness

The receipt-freeness property means that a voter cannot produce a proof of his vote to a third party. In such a security notion, interactions with the third party are allowed before and after the vote. Furthermore, if the vote is performed outside a booth, we can also assume that the third party has access to the channel between the voter and the voting authority: he has knowledge of the transcript, but also of all the information known to the voter, as well as the public information.

A receipt would thus be a proof of the vote v_i , by the voter V_i to a third party: a proof (a witness w') that shows that the bulletin-board contains the vote v_i for voter V_i . The proof must be sound, which means that several proofs are possible, but all for the same statement v_i for a given voter V_i :

Definition 4 (Receipt-Freeness). *A receipt is a witness w' which allows a third party to verify, in an unambiguous way, the vote of a voter $V_i \in \mathbf{V}$:*

$$\exists! v_i, \text{ s.t. } \exists w' \text{ s.t. } R'(\mathbf{B}, V_i, v_i, w') = 1.$$

A voting scheme achieves the receipt-freeness property if there is no such a relation R' , or the witness w' is hard to compute.

3 Incompatible Properties

In this section, we show that a voting scheme cannot provide

- the universal verifiability and the unconditional privacy of the votes, simultaneously, unless all the voters actually vote;
- the universal verifiability and the receipt-freeness, simultaneously, if the transcript of a voter depends on the voter, his vote, his own random, and public values only.

3.1 Universal Verifiability and Unconditional Privacy

Theorem 1. *In the standard model, it is impossible to build a voting scheme that simultaneously achieves the universal verifiability and the unconditional privacy unless all the voters actually vote.*

Proof. Assume we have a *universally verifiable* voting scheme. Then, we want to prove that the *unconditional privacy* cannot be achieved.

Because of the universal verifiability, there exists a public \mathcal{NP} -relation R such that $R(\mathbf{B}, T, \mathbf{V}, w) = 1$, where w is a witness, for a unique tally T and the unique list of voters. Because of the existence of f' , f'' and g , a powerful adversary can guess $\mathbf{V}' = f''(\mathbf{B}')$, $T = f'(\mathbf{B}')$ and $w = g(\mathbf{B}')$ for any $\mathbf{B}' \in \mathcal{L}$: excluding one transcript from \mathbf{B} to build \mathbf{B}' , this adversary can get the name of the excluded voter V' , and the new tally T' , which leaks the vote $v' = T - T'$ of the voter V' .

With an exhaustive search among all the sub-parts of \mathbf{B} , one can then get the vote of a specific voter. \square

This proof strongly relies on the latter sentence. And therefore, the contradiction comes from the above relation R that applies whatever the size of \mathbf{B} is, which allows us to exclude one transcript and use the universal-verifiability relation R .

If the transcripts of *all* the registered voters in L were required in R , the contradiction would not hold anymore, even if it is not clear whether a counter-example exists or not. Anyway, requiring all the registered voters to actually vote is not realistic. A denial of service would become very likely.

In [15], Kiayias and Yung propose a voting scheme in which the privacy is maintained in a distributed way among all the voters. There is no voting authority. They prove that the scheme provides the perfect ballot secrecy which does not correspond to our notion of unconditional privacy: it means that the security of a vote is guaranteed as long as the size of a coalition is not too large and of course according to the tally result and coalition votes. However, in their scheme, each ballot is encrypted using a public-key encryption scheme, that thus requires a computational assumption for the privacy.

In [5], Cramer *et al.* propose a voting scheme that guarantees the unconditional privacy, by using unconditionally secure homomorphic commitments, but only with respect to the voters, and not to the authorities, which would be able to open each individual vote if they all collude.

3.2 Universal Verifiability and Receipt-Freeness

Theorem 2. *Unless private channels are available, the universal verifiability and the receipt-freeness properties cannot be simultaneously achieved.*

Proof. Because of the universal verifiability, v_i is uniquely determined by B_i specific to the voter V_i . Since we exclude private channels, B_i can only be a function of V_i , his vote v_i , some input r_i private to V_i , and public data P_i : $B_i = h(V_i, v_i, r_i, P_i)$. Therefore, r_i is a good witness, and thus a receipt: the scheme is not receipt-free. \square

If the transcript is more intricate, and namely includes some private interactions between the voters and/or the authorities [13], then it may be possible to achieve the two properties simultaneously: B_i is no longer available to the third-party, and thus r_i is no longer a witness either. But such an assumption of private channels is not reasonable in practice.

4 Conclusion

As a conclusion, we have shown that voting systems with usual features cannot simultaneously achieve strong security notions: we cannot achieve simultaneously universal verifiability of the tally and unconditional privacy of the votes or receipt-freeness.

Acknowledgment

This work has been partially funded by the French RNRT Crypto++ Project, and the French ANR 06-TCOM-029 SAVE Project.

References

1. Abe, M., Ohkubo, M.: A length-invariant hybrid mix. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 178–191. Springer, Heidelberg (2000)
2. Aditya, R., Lee, B., Boyd, C., Dawson, E.: An efficient mixnet-based voting scheme providing receipt-freeness. In: Katsikas, S.K., López, J., Pernul, G. (eds.) TrustBus 2004. LNCS, vol. 3184, pp. 152–161. Springer, Heidelberg (2004)
3. Baudron, O., Fouque, P.-A., Pointcheval, D., Poupard, G., Stern, J.: Practical multi-candidate election system. In: Proceedings of the 20th ACM Symposium on Principles of Distributed Computing, pp. 274–283. ACM Press, New York (2001)
4. Benaloh, J., Tuinstra, D.: Receipt-free secret ballot elections. In: Okamoto, T. (ed.) Proceedings of STOC 1994. LNCS, vol. 1976, pp. 544–553. Springer, Heidelberg (1994)
5. Cramer, R., Franklin, M., Schoenmackers, B.: Multi-authority secret-ballot elections with linear work. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 72–83. Springer, Heidelberg (1996)
6. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 103–118. Springer, Heidelberg (1997)
7. Damgård, I., Jurik, M.: A Generalisation, a Simplification and Some Applications of Paillier’s Probabilistic Public-Key System. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer, Heidelberg (2001)
8. Fujioka, A., Ohta, K., Okamoto, T.: A practical Secret Voting Scheme for Large Scale Elections. In: Zheng, Y., Seberry, J. (eds.) AUSCRYPT 1992. LNCS, vol. 718, pp. 248–259. Springer, Heidelberg (1993)
9. Furukawa, J.: Efficient, verifiable shuffle decryption and its requirement of unlinkability. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 319–332. Springer, Heidelberg (2004)
10. Furukawa, J., Sako, K.: An efficient scheme for proving a shuffle. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 368–387. Springer, Heidelberg (2001)
11. Golle, P., Zhong, S., Boneh, D., Jakobsson, M., Juels, A.: Optimistic mixing for exit-polls. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 451–465. Springer, Heidelberg (2002)
12. Groth, J.: A verifiable secret shuffle of homomorphic encryptions. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 145–160. Springer, Heidelberg (2002)
13. Hirt, M., Sako, K.: Efficient receipt-free voting based on homomorphic encryption. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 539–556. Springer, Heidelberg (2000)
14. Jakobsson, M., Juels, A., Rivest, R.: Making Mix-Nets Robust for Electronic Voting by Randomized Partial Checking. In: Proceedings of the 11th Usenix Security Symposium, USENIX 2002, pp. 339–353 (2002)
15. Kiayias, A., Yung, M.: Self-tallying elections and perfect ballot secrecy. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 141–158. Springer, Heidelberg (2002)
16. Kim, K., Kim, J., Lee, B., Ahn, G.: Experimental Design of Worldwide Internet Voting System using PKI. In: SSGRR 2001, L’Aquila, Italy, August 6–10 (2001)
17. Lee, B., Boyd, C., Dawson, E., Kim, K., Yang, J., Yoo, S.: Providing receipt-freeness in mixnet based voting protocols. In: Lim, J.I., Lee, D.H. (eds.) Proceedings of ICICS 2003. LNCS, vol. 2971, pp. 245–258. Springer, Heidelberg (2004)

18. Lee, B., Kim, K.: Receipt-free electronic voting scheme with a tamper-resistant randomizer. In: Lee, P.J., Lim, C.H. (eds.) ICISC 2002. LNCS, vol. 2587, pp. 389–406. Springer, Heidelberg (2003)
19. Neff, A.: A verifiable secret shuffle and its application to e-voting. In: ACM CCCS 2001, pp. 116–125. ACM Press, New York (2001)
20. Ohkubo, M., Miura, F., Abe, M., Fujioka, A., Okamoto, T.: An improvement on a practical secret voting scheme. In: Zheng, Y., Mambo, M. (eds.) ISW 1999. LNCS, vol. 1729, pp. 225–234. Springer, Heidelberg (1999)
21. Okamoto, T.: Receipt-free electronic voting schemes for large scale elections. In: Christianson, B., Lomas, M. (eds.) Security Protocols 1997. LNCS, vol. 1361, pp. 25–35. Springer, Heidelberg (1998)
22. Peng, K., Boyd, C., Dawson, E.: Simple and efficient shuffling with provable correctness and ZK privacy. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 188–204. Springer, Heidelberg (2005)

A Threat Analysis of Prêt à Voter

Peter Y.A. Ryan² and Thea Peacock¹

¹ School of Computing Science, University of Newcastle,
Newcastle upon Tyne, NE1 7RU, United Kingdom

² Department of Computing Science and Communications, University of
Luxembourg 6, rue Richard Coudenhove-Kalergi, L-1359 Luxembourg
`peter.ryan@uni.lu`

Abstract. It is widely recognised that the security of even the best-designed technical systems can be undermined by socio-technical weaknesses that stem from implementation flaws, environmental factors that violate (often implicit) assumptions and human fallibility. This is especially true of cryptographic voting systems, which typically have a large user base and are used infrequently.

In the spirit of the this observation, Karlof et al [11] have performed an analysis of the Chaum [5] and Neff [18] schemes from the “systems perspective”. By stepping outside the purely technical, protocol specifications, they identify a number of potential vulnerabilities of these schemes. In this paper, we perform a similar analysis of the Prêt à Voter [6].

Firstly, we examine the extent to which the vulnerabilities identified in [11] apply to Prêt à Voter. We then describe some further vulnerabilities and threats not identified in [11]. Some of these, such as chain-voting attacks, do not apply to the Chaum or Neff schemes, but are a potential threat in Prêt à Voter, or indeed any crypto system with pre-printed ballot forms. Where appropriate, we propose enhancements and counter-measures.

Our analysis shows that Prêt à Voter is remarkably robust against a large class of socio-technical vulnerabilities, including those described in [11].

1 Introduction

Voting systems are the bedrock of democratic societies, and date back several millennia. While many different mechanisms for voting have been proposed [10], they usually share a similar set of goals such as accuracy, ballot secrecy, verifiability and coercion-resistance [13], [7].

In an attempt to improve the accessibility and efficiency of the election process, democracies have experimented with various automated voting systems that increase speed and accuracy of ballot counting. Arguably, some of these new mechanisms offer greater secrecy, and in the case of remote systems, increased voter participation. However, these attempts have been fraught with problems, many of which are due to reliance on computer hardware and software performing as intended or claimed [3], [2], [12], [14].

Recent proposals for cryptographic voting systems strive to resolve many of these problems by introducing transparency and verifiability. Notable examples are the Chaum [5] and Neff [17, 18] schemes and Prêt à Voter [6]. These strive to provide assurance of secrecy and accuracy without any reliance on the underlying technical system, i.e. software, hardware etc. Instead, the assurance is derived from the properties of the cryptography and a high degree of transparency in the vote recording and counting stages.

While it is important to analyse the core protocol in order to assess its security [4, 13], it is also essential to consider its interaction with the surrounding system, e.g. computer hardware and software, voters and voting officials. In addition, collusion between parties can make the attacks more difficult to detect and resolve. This point was argued by Ryan [21], and more recently, demonstrated by Karlof et al [11] in their examination of the Chaum and Neff schemes from the “systems perspective”.

In this paper, we continue this theme with an analysis of Prêt à Voter, and in doing so, find that it is remarkably robust against many of the vulnerabilities described in [11]. In addition, we identify some new vulnerabilities, and offer mitigation strategies, particularly where they may apply to Prêt à Voter.

The structure of the paper is as follows. In Section 2 we briefly describe the Chaum and Neff schemes. In Section 3, we recall the potential weaknesses identified in [11]. In Section 4, we present an outline of the Prêt à Voter scheme. Section 5 examines the extent to which the attacks of [11] also apply to Prêt à Voter. Following this, in Section 6, we identify further possible attacks and suggest mitigations. In Section 7 some other vulnerabilities of Prêt à Voter are described, along with some counter-measures. Finally, we summarise and conclude in Section 8.

2 The Chaum and Neff Voting Schemes

Although the mechanisms differ in several ways, Prêt à Voter and the Chaum and Neff schemes all provide voter verifiability. More precisely, after a vote is cast, the voter is provided with a physical receipt which encodes the value of her vote, and so ensuring secrecy. The voters can later check that their ballot receipt has been correctly posted to the *Web Bulletin Board* (WBB). Tabulation is performed by decryption and anonymisation via robust anonymising mixes. To ensure accuracy, various checking mechanisms are deployed to detect any failure or corruption in either the encryption or decryption of the receipts.

Due to space constraints, we only summarise the key features of the Chaum and Neff schemes, and refer the reader to [6, 4, 18, 19] for details, or [13, 23] for overviews.

In the booth, the voter interacts with a machine, during which the voter’s choice is communicated to the machine and encoded in a receipt. The voter is given the opportunity to verifying that their vote is correctly encoded. This is achieved by way of a “cut-and-choose” protocol, in which the device commits to two or more encryptions of the vote, one of which the voter chooses to retain.

The remaining encryptions are then opened, verified and discarded. A digital copy of the chosen receipt is retained by the device and, once the voting period has ended, is posted to a WBB. The voter retains a hard copy of her encrypted ballot receipt, which she can subsequently check on the WBB to make sure that the receipt has been correctly recorded.

The receipts are shuffled and decrypted in a series of anonymising mixes to ensure that no link remains between the encrypted ballot receipts and the final decrypted vote values. The results of each stage of the mix process are posted to the WBB.

Aside from the details of the cryptographic primitives involved and the mix processes, the essential difference between the two schemes is in the format of the receipt.

In the Neff scheme, the receipt consists of a matrix of El Gamal ciphertexts of the digits “0” or “1”. A chosen candidate is distinguished by the arrangement of digits in the corresponding row.

In the Chaum scheme uses the RSA algorithm and the voter’s choice is represented using visual cryptography [16] to generate a 2-D ballot image showing the candidate name, which is split between two encrypted, transparent layers. The ballot image is visible when the two sheets are accurately overlaid but, when separated, each sheet is a pattern of random pixels. The voter selects one layer to retain as a receipt, so without knowledge of the crypto keys, the receipt does not reveal the voter’s choice.

3 Cryptographic Voting Protocols: Chinks in the Armour

The vulnerabilities considered in [11] fall into three main categories: those due to subliminal channels, “social engineering”-style attacks against the cryptographic protocols and denial of service attacks. In this section, we briefly recall each of these vulnerabilities and how they may apply to the Chaum and Neff schemes.

3.1 Subliminal Channels

Subliminal channels provide a means for a malicious agent to transmit information over a channel in a way that is hidden from the legitimate users of the channel. They can arise whenever there are alternative valid encodings of the “intended” information. Additional information can be encoded in suitable choices between these alternatives. Public access to the WBB makes this a particularly virulent threat for voter-verifiable schemes. There are two classes of subliminal channel identified in [11]: *random* and *semantic*.

The Neff scheme makes use of randomised crypto variables in the creation of the ballot receipt giving rise to a possible *random* subliminal channel. By judicious selection of random values, the voter’s choice could be encoded in the encrypted receipt [11]. Any agent with knowledge of the coding strategy could then gather this information by observing the posted receipts.

Random channels do not occur in the Chaum scheme, as it uses deterministic algorithms. However, *semantic* subliminal channels, can occur if there are

alternative valid representations of the ballot image. Certain information could then be conveyed by altering the ballot image, for example by adjusting the font size or positioning of the image. Note that, in contrast to the random channel of the Neff scheme, this channel emerges after the anonymising mixes, when the decrypted ballot images emerge. Thus a malicious device would presumably encode information about the voter identity rather than the vote value.

Mitigation. Counter-measures for random subliminal channels are tricky, since the randomness may be essential for security properties. A possible approach, attributed to Neff [11], is to require the use of pre-determined randomness, e.g., from pre-generated tapes. In effect, the non-determinism is resolved before the voter’s choices are communicated to the system. The difficulty with this approach is ensuring that the devices adhere to this pre-determined entropy. In personal communication, Neff describes a recent implementation of his scheme, in which ballots are created in advance by a multi-authority process similar to a mix-net. The function of the device is completely deterministic, so there is no possibility of a subliminal channel. In addition, the voter only makes one random choice for the entire ballot, rather than for each option, as described in [11]. Details of these innovations can be obtained from [1].

With the Chaum scheme, one could enforce a standard format for each the ballot image for each candidate option. This also runs into the difficulty of monitoring and enforcing adherence. It is also difficult to square with the possibility of “write-ins” that the scheme enables. In personal communication, Chaum observed that such semantic channels are not strictly subliminal: exploitation of such channels would be detectable.

Another possibility is to use trusted hardware [11], but this is contrary to the principle of transparency and minimal dependence which the Chaum, Neff and Prêt à Voter (see later) schemes aim to achieve.

3.2 Social Engineering Attacks

Both the Chaum and Neff schemes require non-trivial interactions between the voter and the machine. Both schemes involve “cut-and-choose” protocols, and the sequence of steps in the protocol can be highly significant. These are designed to detect any attempt to construct receipts that do not encode the voter’s true choice. We will refer to the choice the voter makes in a “cut-and-choose” step as the “protocol choice” to distinguish this from the voter’s candidate choice.

By re-ordering the steps in the protocol, or introducing extra ones, the machine could learn the voter’s protocol choice before it has to commit to the receipt encoding [11]. In this case the device can corrupt the vote value with impunity. Voters may not notice or appreciate the significance of such changes in the protocol execution.

Similarly, the machine could feign an error and re-boot after it learns the voter’s protocol choice. Relying on the voter making the same choice in the second round of the protocol, the machine then constructs a receipt for a different candidate. If the voter changes her mind, the machine re-boots again until the voter gets it “right” [11].

Another possible vulnerability of the Chaum scheme, not identified in [11], but mentioned in [21], is as follows. The machine attempts to corrupt the vote value by incorrectly constructing one of the layers. If the voter chooses the other layer for retention, the corruption will go undetected. However, if the voter chooses the corrupted layer (which would lead to detection), the machine could try to fool the voter by printing “destroy” instead of “retain” on the layer that the voter chose as the receipt. If the voter fails to notice this, or simply ignores it, the corruption will again pass undetected. This is another way that a corrupt machine could undermine the “cut-and-choose” element of the protocol. Even if the voter does notice the switch and is confident that they are right, it may be difficult to demonstrate this to a voting official.

Mitigation. We refer the reader to [11] or [23] for suggested mitigations. The obvious approach is to try to ensure that voters understand the procedures and appreciate their motivation. This is similar to the notion of instilling a “security culture” in an organisation. In practice of course, this may not really be feasible. In the context of an election system, where the set of users can be vast and usage infrequent, the possibilities for voter education is limited.

An alternative solution is to make the voter experience much simpler, as with Prêt à Voter. However such simplicity seems to be at the cost of needing to place more trust in the system.

3.3 Denial of Service

There are several ways in which DoS attacks could disrupt or invalidate an election. See [11] for a discussion. For all such attacks, adequate error-handling and recovery strategies need to be in place. In addition, some form of back-up is desirable, e.g. a *voter-verifiable paper audit trail* (VVPAT) [15]. We return to these attacks and counter-measures later, when we examine the robustness of Prêt à Voter.

4 The Prêt à Voter Scheme

We now present an overview of the Prêt à Voter scheme. For full details see [6]. Prêt à Voter is inspired by the Chaum scheme, but replaces the visual cryptographic techniques with a conceptually and technologically simpler mechanism to represent the encrypted vote value in the ballot receipt. In the polling station, voters select a ballot form at random, an example of which is shown below.

Obelix	
Idefix	
Asterix	
Panoramix	
	7r.J94K

In the booth, the voter makes her selection in the usual way by, for example, placing a cross in the right-hand (RH) column against the candidate of choice. She now separates the (RH) and left-hand (LH) sides, and the latter, which carries the candidate list, is discarded to leave the ballot receipt. Supposing a vote for Idefix, the receipt would appear as follows:

X
7rJ94K

She then leaves the booth and authenticates herself with an official, who scores off her name in the register. The receipt is placed under an optical reader or similar device, to record the cryptographic value at the bottom of the strip, and the numerical representation of the cell into which the cross has been entered. The voter retains the hard copy of the RH strip as her receipt. An anti-counterfeiting device would be incorporated, and the receipt would be digitally signed when the vote is cast.

Possession of a receipt might appear to open up the possibility of coercion or vote-buying. However, the candidate lists on the ballot forms are randomised, so the order in which the candidates are shown is unpredictable. Clearly, as long as the LH strip is removed, the RH strip alone does not indicate which way the vote was cast.

The crypto value printed on the bottom of the receipt, the “onion”, is the key to extraction of the vote. Buried cryptographically in this value, is the seed information needed to reconstruct the candidate list. This information is encrypted under the secret keys of a number of tellers. Thus, only the tellers acting in consort are able to reconstruct the candidate order and so interpret the vote value encoded on the receipt.

Once the election has closed, all the receipts are transmitted to a central tabulation server which posts them to a secure WBB. This is an append-only, publicly visible facility. Only the tabulation server can write to this and, once written, anything posted to it will remain unchanged. Voters can visit this WBB and confirm that their receipt appears correctly.

After a suitable period, the tellers take over and perform a robust, anonymising, decryption mix on the batch of posted receipts. Various approaches to auditing the mixes can be used to ensure that the tellers perform the decryptions correctly. Details of this can be found in [6].

Another place where the accuracy of the scheme could be undermined is in the vote capture stage, and the encoding of votes in the receipts. In the case of Prêt à Voter, this could occur if the ballot forms are incorrectly constructed, i.e., the candidate list shown does not correspond to the crypto seeds buried in the onion value. Various mechanisms can be deployed to detect and deter such corruption. The approach suggested in [6] is to perform a random pre-audit of the

ballot forms. In other words, both independent third parties and the voter can check the well-formedness of the ballot forms. The checks performed in each case, however, differ. The former involves extracting the crypto seeds, re-computing the onions and checking that they correspond to the candidate permutation.

Later in this paper, we discuss an alternative approach using on-demand creation and printing of forms along with a “cut-and-choose mechanism” and post-auditing.

For full details of the mechanisms used to detect any malfunction or misbehaviour by the devices or processes that comprise the scheme, see [6].

5 Systems-Based Analysis of Prêt à Voter

In this section we examine the extent to which the vulnerabilities identified in [11] apply to Prêt à Voter.

5.1 Subliminal Channels

Subliminal channels of the type identified by Karlof et al, are not a problem for Prêt à Voter. This is due mainly to the rather special and, to our knowledge unique, way that votes are encoded in Prêt à Voter. Most cryptographic voting schemes require the voter to supply her vote choice to the device, which then produces a (verifiable) encryption. In the case of Prêt à Voter, the voter’s choice is encoded in a randomised frame of reference. It is the information that allows this frame of reference to be recovered, the “seed” value, that is encrypted. This can be done ahead of time without needing to know the vote value. In Prêt à Voter, the ballot forms are generated in advance and allocated randomly to voters. Hence, the cryptographic commitments are made before any linkage to voter identities or vote choices are established.

Regarding semantic subliminal channels, suitable implementation should ensure that, for a given ballot form and vote choice, the digital representation in a Prêt à Voter receipt is unique, i.e., the onion value and the index value indicating the chosen cell on the LH column. Hence, there should be no possibility of a semantic subliminal channel.

Like the Chaum scheme, Prêt à Voter “Classic”, [6], uses deterministic cryptographic algorithms. In fact, in both schemes, all the tellers’ operations can be made deterministic: encryption and decryption are deterministic and the tellers can be required to post batches of transformed ballot receipts in lexical order at each stage, thus eliminating random subliminal channels. It is not clear whether this is really necessary, as the tellers do not have access to any privacy sensitive information.

5.2 Social Engineering Attacks

In Prêt à Voter, the voter does not engage in a multi-step protocol with the device so there is little scope for any social engineering-style attacks.

It is worth noting that in Prêt à Voter, the analogue of the “cut-and-choose” element of the Chaum or Neff schemes is performed by independent auditing authorities who check a random selection of the ballot forms. This removes the need for a “cut-and-choose” step in the voter’s phase of the protocol, as the authority commits to the crypto material on the ballot forms ahead of the election. A random selection of these are checked by the auditors for well-formedness. This can be done before, during and after (on unused, left-over forms) the election period. Assuming that they pass the checks, audited forms are destroyed. Forms that fail the checks would be retained for forensic purposes.

Prêt à Voter also allows the possibility of voters performing random checks on the ballot forms in addition to checks performed by auditors. This is more in the spirit of arranging for the trust to reside solely in the voters themselves. Care has to be taken, however, to avoid introducing other vulnerabilities, such as the possibility of checking ballot forms that have been used to cast votes.

5.3 Denial of Service

Whilst these schemes succeed in removing the need to trust the devices and tellers for the accuracy requirement, we may still be dependent on them to some extent for availability. Unless suitable measures are taken, the failure of a teller for example, could at least hold up and in the worst case block the tabulation. Corruption of the digital copies of receipts would call the election into doubt. Thus measures must be taken to make the scheme robust against (manifest) failure or corruption of devices. In other words, we have ensured that, with high probability, any (significant) failures or corruption will be detected, but we still have to address the issue of error handling and recovery.

A possible enhancement of Prêt à Voter, detailed in [24], is to replace the decryption mixes with re-encryption mixes. This has a number of advantages, one being that recovery from DoS failures is much easier. There are a number of reasons for this:

- The mix tellers do not need secret keys, they simply re-randomise the encryption. A failed mix teller can therefore simply be replaced without all the unpleasantness of having to surgically extract keys.
- The mix and audit can be independently re-run. With (deterministic) decryption mixes, the selection of links for audits cannot be independently selected on the re-run of the mix without compromising secrecy.

The use of threshold encryption schemes would also help to foil DoS attacks by ensuring the failure of a proportion of the (decryption) tellers could be tolerated.

In [20], it is suggested that an encrypted VVPAT [15]-style mechanism be incorporated. As the device scans the voter’s receipt, it generates an extra copy. Once this copy has been verified by the voter (and possibly an official), it is entered into a sealed audit box. This provides a physical back-up of receipts cast, should recovery mechanisms need to be invoked.

We emphasise that this is actually quite different to the conventional notion of VVPAT that generates a paper audit trail of unencrypted ballot slips. The

conventional form of VVPAT suffers from a number of privacy problems. For example, if the trail retains the order of votes cast, receipts could be linked to voters by comparing the order in which votes are cast with that of voters entering the booth. Also, any mismatch between the voter's choice and the paper audit record can be difficult to resolve without compromising the voter's privacy.

Where encrypted receipts are recorded, these problems disappear. Note that, due to the encrypted form of the receipts, it is possible for monitors to verify a vote as it is cast, in addition to the voter checking. Any discrepancies can be resolved without loss of voter privacy. We will henceforth refer to such a mechanism as *Verified Encrypted Paper Audit Trail* (VEPAT).

5.4 Discarded Receipts

As with the Chaum and Neff schemes, carelessly discarded receipts could be a problem in Prêt à Voter, as this could indicate which receipts will not be checked by voters on the WBB [11]. Malicious parties could delete or alter the corresponding receipts, confident that the voters will not check them on the WBB.

Aside from voter education [11], a possible mitigation is, again, to invoke a VEPAT mechanism, as described above. Independent observers could check the correspondence between the VEPAT and the contents of the WBB. This has the added advantage in that less reliance need be placed on the voters' diligence in checking their receipts on the WBB.

5.5 Invalid Digital Signatures

In the Chaum scheme, digital signatures act as a counter-measure against faked receipts being used to discredit election integrity. However, a device that falsified signatures could be used to discredit voters, leaving them without a way to prove a dishonest system [11].

Voters should thus be provided with devices capable of verifying the digital signatures. Such devices could be provided by various independent organisations, such as the Electoral Commission, etc. Similar measures could be utilised for Prêt à Voter.

Given that encrypted receipts can be cast in the presence of officials and other observers, we have the possibility of checking digital signatures at the time of casting and applying physical authentication mechanisms, such as franking, to the receipt.

5.6 Insecure Web Bulletin Board

Like the Chaum and Neff schemes and indeed many cryptographic voting schemes, Prêt à Voter relies on a secure WBB to allow voter and verifiability. In a possible attack, the WBB arranges for the voter to see a correct record of her ballot receipt, which, in collusion with the mix-net, has been deleted or altered. As a result, the voter could mistakenly believe that her vote has been accurately counted [11].

However, we note that suggested mitigations, such as robust data storage and allowing only authorised write access to the WBB [11] are still vulnerable to corruption. The challenge is provide a trusted path from the WBB to the voter. The problem of implementing secure web bulletin boards is the subject of ongoing research in the cryptographic voting community.

6 Further Vulnerabilities

In this section, we discuss other possible vulnerabilities not identified in [11], and suggest appropriate mitigation strategies.

6.1 Doll Matching Attack

This vulnerability is specific to the Chaum scheme and is not identified in [11]. Each of the layers that combine to reveal the ballot image have to carry a pair of “dolls”, analogues of our “onions”, one for each layer. It is essential that these are identical between the layers. The voter is required to check that values on the two layers match. Failure to do this could allow the device to construct fake receipts without detection. It might seem difficult to produce a fake “doll” that alters the vote value whilst differing from the real “doll” in a way that is visually almost imperceptible. This would be analogous to finding (approximate) collisions of a cryptographic hash function. Nevertheless, this should still be considered a potential vulnerability.

A counter-measure is to ensure that visual matching of the dolls is as easy to perform as possible. Thus, for example, the dolls might be encoded aligned bar codes on the two layers. Any mismatch would then show up as a misalignment of the bars.

6.2 Undermining Public Confidence in the Secrecy of Encrypted Receipts

Another potential attack against schemes employing encrypted receipts, is as follows. The Mafia (falsely) claim to have a way of extracting a vote from the encrypted receipt. If a sufficient number of voters were convinced by such a claim, and so influenced to alter their vote, it may be possible to undermine the outcome of the election. For instance, a coercer might urge voters to submit their receipts, claiming that the “correctness” of the choice would be checked. Some reward, such as entry into a lottery, would be offered for receipts that passed the supposed checks.

Countering such a psychological attack, other than by voter education, could be difficult.

6.3 Side-Channel Attacks

Karlof et al do not discuss the possibility of the vote-capture devices leaking the voters choices via side-channels, e.g., hidden wires, wireless-enabled devices, etc.

This may be because they are regarded as inevitable. In fact, in the case of Prêt à Voter, such channels are not, in fact, a problem.

As explained earlier, the voter's choice does not need to be communicated to the device in order to obtain the encoding of this choice in the receipt. Hence, even if such channels existed, they could not be exploited to leak information from the booth device.

However, the potential for other kinds of side-channels still exists: hidden cameras, "invisible" dots on the receipts etc.

6.4 Kleptographic Channel Attacks

The current version of Prêt à Voter is, however, vulnerable to kleptographic channels, another form of subliminal channel. These were originally described by Young and Yung [25] and their relevance to voting schemes identified by Gogolewski et al [9]. In the case of Prêt à Voter, the idea is that the authority creating the ballot forms would carefully select the seed values in such a way as to encode information about the candidate list in the onion values. This encoding would use some secret key shared with a colluding third party. Thus, seeds would be chosen so that a certain keyed hash applied to the onion value would carry information about the corresponding candidate order. Clearly this would require a significant amount of searching in the seed space and computation, but the resulting selection of seeds would appear random to anyone not knowing the coding strategy and secret hash key.

It is fairly straightforward to eliminate this kind of attack by arranging for the seeds to be created in a distributed fashion by several entities in such a way that no single entity can control or know the resulting seed values. Ryan et al [24] describes such a mechanism, in which several trustees create proto-ballot forms in a kind of pre-mix. The resulting proto-ballot forms have two distinct onions encrypting the same seed value. The seed value, and hence candidate list, can be extracted from one of these on demand to reveal the full ballot form. Full details can be found in [24].

7 Prêt à Voter Specific Vulnerabilities

Our analysis of Prêt à Voter revealed other possible vulnerabilities that are specific to the scheme. We discuss them and suggest possible mitigations.

7.1 Chain Voting

Chain voting is a well known style of attack that can be effective against some conventional paper ballot schemes. In this attack, the coercer smuggles an unused ballot form out of the polling station and marks his preferred candidate. The voter is told that they will be rewarded if they emerge with a fresh, unmarked form. This can then be marked again and passed to the next voter.

Particularly vulnerable are election systems in which, as in the UK, the ballot forms are a controlled resource: on registration, voters are given one ballot form and they are observed to cast this before existing the polling station. The voter is then under duress to cast the form marked by the coercer and to retain the fresh form that they are provided with when they register. The procedures arguably make it harder for the coercer to initialise the attack. However, a determined attacker would certainly find a way, for example by bribing an official, or placing a fake form in the ballot box. Once the attack is initialised, the procedure works in the coercer’s favour.

A possible counter-measure is to make ballot forms freely available in the polling stations, as, for example, in French elections. Voter identity is checked when casting a vote rather than at the time of collecting a ballot form. Thus, as it is not certain that a marked form was actually used to cast a vote, the motivation for the attack is undermined.

Neither the Chaum nor the Neff schemes, in which the ballot forms and receipts are generated on demand in the booth, are vulnerable to this style of attack. Prêt à Voter is, however, potentially vulnerable, as the ballot forms are pre-printed. The counter-measure above does not work as cast receipts are posted to a publicly-verifiable WBB, so allowing the coercer to confirm whether or not the designated ballot form was actually used.

7.2 Mitigation

Observe that at the time of making her candidate choice, it is only necessary for the voter to see the candidate list. A possible counter-measure therefore is to conceal the onion by a “scratch strip”, similar to that used in lottery tickets. The procedure could then be for the voter to register and collect a fresh ballot form, with scratch strip intact. The voter goes to the booth, marks her selection, then detaches and destroys the LH strip. She exits the booth and takes her receipt to an official who checks her identity and that the scratch strip is intact. The voter, or an official, now removes the strip and records the receipt as previously described.

Steps would need to be taken to ensure that the scratch strips cannot be scanned with some device to read the concealed onions. This has reportedly been done using the laser photo-acoustic effect. See [8] for details. Although expense and technical know-how would be required on the part of the attacker, it should still be considered a possible threat.

A rather different counter-measure is to return to an on-demand creation of ballot forms, e.g. printing in the booths. The scheme for the distributed generation of encrypted ballot forms of [24] could be used here. This avoids chain voting and certain chain of custody issues but at the cost of having to re-introduce the voter involvement in the “cut-and-choose” along with post-auditing to the protocol. The trade-offs involved in this are investigated in [22].

7.3 Authority Knowledge

In the current version of Prêt à Voter, the authority has knowledge of ballot form information, i.e. the crypto seeds used to generate the candidate offsets, hence

the onions, and, in particular, the association between these values. This means that the authority has to be trusted not to leak this information. Even if the authority is entirely trustworthy, there is always a danger of this information being leaked during distribution or storage of the ballot forms, i.e., chain of custody issues.

A possible solution is to arrange for the ballot form material to be constructed in a distributed fashion, in such a way as to ensure that no single entity knows the association of onions and candidate lists. As noted previously, up until the time of casting a vote, it is not necessary for the voter to see the onion. One could thus devise a scheme in which the onion and candidate list are never exposed simultaneously.

We now outline a simple scheme for distributed construction of “scratch card” style ballot forms. It is based on onions encrypted using El Gamal, or a similar randomised encryption algorithm. The ballot form material is generated by a number of ballot clerks. The first clerk generates a large quantity of onions, which then enter a re-encryption pre-mix involving the other clerks. The last clerk collects the resulting permuted, random onions and, for each one, produces two re-encryptions. These paired onions are now printed onto ballot forms, one at the bottom of the LH column, the other on the bottom of the RH column. A proto-ballot form is shown below:

	7rJ94K
jH89Gq	

These two onions should correspond to the same candidate list. The RH onion is now concealed with a scratch strip.

These are now shuffled and passed to a final clerk who then dispatches the visible, LH onions to the tellers. The tellers send back the corresponding candidate list which is now printed in the LH column and the LH onion is removed. This results in ballot forms similar to those proposed in Section 4, but with the onion value concealed by a scratch strip.

Note that no single entity now knows the association of onion and scratch strip. Strictly speaking, the last two clerks acting in collusion could form the association, but the scheme can be elaborated to raise the collusion threshold. Interestingly, we note also that, even though these two clerks in collusion know the onion/candidate list association, they cannot prove this knowledge to a third party as they do not know the necessary crypto seeds used to compute the onion values. Ballot forms can be randomly audited as before.

Alternatively, the pre-mix approach of [24] alluded to earlier as a counter to kleptographic attacks, would also be effective here to eliminate the authority knowledge problem.

7.4 Enforcing the Destruction of the Left-Hand Strips

After the voter’s selection has been marked on the ballot form, the left-hand strip must be destroyed. Failure to do so would allow the voter to use it as proof of her vote to a third party. Clearly, this would lay the system open to coercion.

Several ways of enforcing this are possible. The voter could be required to destroy the left-hand strip in the presence of an official, preferably in some mechanical shredding device. This could be done at the time of casting the ballot form, as suggested above. However, care would have to be taken to ensure that the official is not able to record the association of the receipt and candidate list.

Another possibility is to have devices in the booth that would automatically cut off and destroy the LH strip and then pass the receipt into a scanner. This would make the voter’s interaction simpler, but such devices would have to be carefully evaluated, and run counter to the “trust nobody and nothing” philosophy of voter-verifiable schemes.

Another possibility is to make “decoy” left-hand strips freely available in the booths, so the voter cannot convince the coercer that the one she emerges with is genuine.

7.5 Confusion of Teller Modes

As previously mentioned, the tellers perform an anonymising decryption mix on the receipts posted to the WBB. However, they also have a role in checking the construction of ballot forms, both by auditors and, potentially, voters [6]. For ballot forms selected for audit, the onions are sent to the tellers, who return the corresponding seed values. The auditors then re-compute the onion values and candidate offsets, and check that they are correct. In voter checking, the tellers return the candidate ordering corresponding to the onion value sent by the voter.

The checked forms should then be discarded. If the audited forms were later used to cast a vote, there could be a threat to ballot secrecy. Conversely, it should not be possible to run a check on a form that has been used to cast a vote.

To mitigate this, ballot forms could be checked by voters in the presence of an official, who then ensures that used forms are discarded. Forms could be invalidated once used, for example, using the described scratch strip mechanism. An authentication code could be overprinted on the scratch strip that would be necessary to enable the checking mode. Revealing the onion would entail removing the scratch strip and the code along with it, ensuring that the form could not be reused later. Alternatively, an authorisation code could be introduced on the LH strip that would be destroyed at the time of casting.

8 Conclusions

In this paper, we have performed a threat analysis of the Prêt à Voter scheme. In particular, we have extended the analysis of Karlof et al with some further systems based vulnerabilities not identified in [11], and considered the applicability of these vulnerabilities to the Prêt à Voter scheme.

Prêt à Voter has proved to be remarkably resilient to these vulnerabilities, many of which stem from voter interaction with devices and generation of entropy at the time of voting. However, Prêt à Voter, is potentially prey to chain voting attacks, which do not apply to schemes in which the crypto material is generated on demand. In fact, as we have discussed, this attack is particularly virulent in the context of voter-verifiable schemes with pre-prepared ballot forms and Web Bulletin Boards. Wherever such threats apply to Prêt à Voter, counter-measures have been suggested.

As with any secure system, voting schemes require great care in their design and evaluation, not only of the cryptographic core, but also of the surrounding system. This analysis has provided valuable insight into the way forward for Prêt à Voter, and some enhancements have been suggested. It has also underlined the need for adequate error-handling and recovery strategies, and that a VPEAT style mechanism would be highly desirable.

The analysis presented here, as with the analysis of Karlof et al, does not of course constitute an exhaustive, systematic, identification of all the system-based threats to Prêt à Voter. Arguably, complete coverage for such an analysis could never be guaranteed given the open-ended nature of systems. However, we feel that this analysis constitutes a useful first step towards a more systematic analysis technique for crypto voting systems. Here, we explore the space of possible failure modes and adversary capabilities, which will enable us to build a threat model.

We already have the start of a taxonomy of attacks, i.e., classification into subliminal channels, side-channels, kleptographic channels, social engineering threats, psychological etc. It seems likely that a form of design-level information flow analysis should help guide further analysis. This will be pursued in future research.

Finally, we conclude that, provided that, in addition to a formal analysis of the core, technical system, socio-technical considerations are incorporated into the design and evaluation phases, there is every reason to suppose that cryptographic schemes of this kind can provide trustworthy, verifiable elections.

Acknowledgements

The authors would like to thank Michael Clarkson, David Chaum, Michael Jackson, Steve Kremer, Andy Neff, Andrey Povyakalo, Mark Ryan and Luca Vigano for fruitful discussions and DSTL and the EPSRC DIRC project for partial funding of this work.

References

1. Votehere, <http://www.votehere.net/default.php>
2. The trouble with technology. *The Economist*, September 16 (2004)
3. Bannet, J., Price, W., Rudys, A., Singer, J., Wallach, D.: Hack-a-vote: Security issues with electronic voting systems. *IEEE Security and Privacy* 2(1) (January/February 2004)

4. Bryans, J., Ryan, P.Y.A.: A dependability analysis of the chaum voting scheme. Technical Report CS-TR-809, University of Newcastle upon Tyne (2003)
5. Chaum, D.: Secret-ballot receipts: True voter-verifiable elections. *IEEE Security and Privacy* 2(1), 38–47 (2004)
6. Chaum, D., Ryan, P.Y.A., Schneider, S.: A practical, voter-verifiable election scheme. In: di Vimercati, S.d.C., Syverson, P.F., Gollmann, D. (eds.) *ESORICS 2005*. LNCS, vol. 3679, pp. 118–139. Springer, Heidelberg (2005)
7. Fujioka, A., Okamoto, T., Ohta, K.: A practical secret voting scheme for large scale elections. In: *Workshop on the Theory and Application of Cryptographic Techniques: Advances in Cryptology*, pp. 244–251. ACM, New York (1992)
8. Gerck, E.: Instant lottery cards too, re: reading pins in ‘secure’ mailers without opening them (2005), <http://www.mail-archive.com/cryptography>
9. Gogolewski, M., Klonowski, M., Kubiak, P., Kutylowski, M., Lauks, A., Zagorski, F.: Kleptographic attacks on e-election schemes with receipts. In: Müller, G. (ed.) *ETRICS 2006*. LNCS, vol. 3995, pp. 494–508. Springer, Heidelberg (2006)
10. Jones, D.W.: A brief illustrated history of voting (2003), <http://www.cs.uiowa.edu/~jones/voting/pictures>
11. Karlof, C., Sastry, N., Wagner, D.: Cryptographic voting protocols: A systems perspective. In: *USENIX Security Symposium* (2005)
12. Kohno, T., Stubblefield, A., Rubin, A.D., Wallach, D.S.: Analysis of an electronic voting system. In: *Symposium on Security and Privacy*. IEEE, Los Alamitos (2004)
13. Kremer, S., Ryan, M.: Analysis of an electronic voting protocol in the applied pi-calculus. In: Sagiv, M. (ed.) *ESOP 2005*. LNCS, vol. 3444, pp. 186–200. Springer, Heidelberg (2005)
14. Lauer, T.W.: The risk of e-voting. *Electronic Journal of e-Government* 2(3) (December 2004)
15. Mercuri, R.: A better ballot box? *IEEE Spectrum Online* (October 2002)
16. Naor, M., Shamir, A.: Visual cryptography. In: De Santis, A. (ed.) *EUROCRYPT 1994*. LNCS, vol. 950, pp. 1–12. Springer, Heidelberg (1994)
17. Neff, A.: A verifiable secret shuffle and its application to e-voting. In: *Conference on Computer and Communications Security*, pp. 116–125. ACM, New York (2001)
18. Neff, A.: Practical high certainty intent verification for encrypted votes (2004), <http://www.votehere.net/documentation/vhti>
19. Neff, A.: Verifiable mixing (shuffling) of el-gamal pairs (2004), <http://www.votehere.net/documentation/vhti>
20. Randell, B., Ryan, P.Y.A.: Voting technologies and trust. *IEEE Security & Privacy* (2005) (to appear)
21. Ryan, P.Y.A.: Towards a dependability case for the chaum voting scheme. In: *DIMACS Workshop on Electronic Voting – Theory and Practice* (2004)
22. Ryan, P.Y.A.: Putting the human back in voting protocols. In: Christianson, B. (ed.) *Security Protocols 2006*. LNCS, vol. 5087, pp. 20–25. Springer, Heidelberg (2009)
23. Ryan, P.Y.A., Peacock, T.: Prêt à voter: a systems perspective. Technical Report CS-TR-929, University of Newcastle upon Tyne (2005)
24. Ryan, P.Y.A., Schneider, S.A.: Prêt à voter with re-encryption mixes. Technical Report CS-TR-956, University of Newcastle upon Tyne (2006)
25. Young, A., Yung, M.: The dark side of “Black-box” cryptography, or: Should we trust capstone? In: Kobitz, N. (ed.) *CRYPTO 1996*. LNCS, vol. 1109, pp. 89–103. Springer, Heidelberg (1996)

Anonymity in Voting Revisited

Hugo Jonker^{1,2} and Wolter Pieters^{3,*}

¹ Eindhoven University of Technology, The Netherlands

² University of Luxembourg, Luxembourg

h.l.jonker@tue.nl, hugo.jonker@uni.lu

³ University of Twente, The Netherlands

w.pieters@utwente.nl

Abstract. According to international law, anonymity of the voter is a fundamental precondition for democratic elections. In electronic voting, several aspects of voter anonymity have been identified. In this paper, we re-examine anonymity with respect to voting, and generalise existing notions of anonymity in e-voting. First, we identify and categorise the types of attack that can be a threat to anonymity of the voter, including different types of vote buying and coercion. This analysis leads to a categorisation of anonymity in voting in terms of a) the strength of the anonymity achieved and b) the extent of interaction between voter and attacker. Some of the combinations, including weak and strong receipt-freeness, are formalised in epistemic logic.

1 Introduction

In the field of peer-to-peer (P2P) networks, much effort has been put into formalizing the concept of anonymity of messages (e.g. [14]). Intuitively, anonymity means that it is impossible to determine who sent which message to whom. Depending on the context, different formalizations of the notion of anonymity seem to be necessary [7].

The concept of anonymity is also of importance in electronic voting – often, voters should have the ability to vote without anybody else knowing which option they voted for (although in some countries, such as the United Kingdom and New Zealand, this is ultimately not the case). In the electronic voting community, the property expressing precisely that is usually called “privacy” instead of anonymity [6]. In voting, however, enabling privacy is not sufficient, as this does not prevent vote buying. To prevent vote buying, an election needs to require privacy – no voter should be able to convince any other party of how she voted.

The concept of receipt-freeness expresses that a voter cannot convince any other party of how she voted by creating a receipt. The notion has been introduced by [2], after which various receipt-free voting protocols were proposed, such as [9,17]. Delaune et al. [5] provide a definition of receipt-freeness based on observational equivalence. Independently, Jonker and De Vink [10] provide an

* Research was carried out partially at Radboud University, Nijmegen, The Netherlands, and supported by NWO, the Netherlands Organisation for Scientific Research.

alternate definition that allows identification of receipts. Juels et al. note in [11] that receipt-freeness is not sufficient to prevent coercion in electronic elections, and they introduced the notion of coercion-resistance. This broader notion is again formalized by Delaune et al. in [6].

Given the differences in approaches and in notions, the question arises whether these notions capture the specific needs for anonymity in voting. The three main levels of anonymity that have been identified in voting, capture progressively more strict notions of anonymity. The notion of receipt-freeness was motivated as necessary to provide secret-ballot elections. If receipts can be obtained, using a voting booth makes no difference to the secrecy: Votes can be bought, and voters can be coerced.

To address the question of whether or not the notion of receipt-freeness is sufficient, we reexamine voter influencing, focusing on vote buying. What is vote buying, when can an action be called vote buying and when is it an election promise? As this is, ultimately, a subjective issue, the goal is not to provide a yes-or-no test. Instead, we aim to arrive at a characterisation of vote buying / election promises, which will enable election officials to decide which practices are allowed and which should be abolished. Based on these findings, we then reexamine the concept of receipt-freeness and adapt it to encompass uncovered issues.

1.1 Related Work

Distinctions between vote buying and election promises have been investigated by economists, philosophers and political scientists before.

Van Acker [1] discusses the relation between the notions of coercion, forced abstention, randomisation and simulation. However, he includes vote buying in the concept of coercion.

Kochin and Kochin [12] discuss the issue of giving benefits to individual voters versus giving benefits to identifiable groups. They also consider the difference between benefits offered through the normal processes of government (related to being elected) versus benefits offered through private arrangements. Thirdly, they mention that trading votes for or against proposals between parties or members in parliament is acceptable.

The latter practice is also mentioned by Hasen [8] and called “legislative logrolling”. Hasen further differentiates the issues of corporate vote buying, payments to increase turnout, campaign promises and campaign contributions, and vote buying in so-called “special district” [1] elections.

Schaffer [18] distinguishes between *instrumental*, *normative* and *coercive* compliance in relation to vote buying. Instrumental compliance covers tangible benefits in exchange for votes, normative compliance means voting based on a feeling of obligation, and coercive compliance denotes voting based on threats. Schaffer also mentions the possibility that money is offered for *not* changing voting behaviour. In order to check compliance, a buyer may monitor the individual vote,

¹ “A special purpose unit of government assigned the performance of functions affecting definable groups of constituents more than other constituents”.

monitor the aggregate turnout, prevent people from voting altogether, make the rewards dependent on his election, make voters believe in his goodness or make voters feel personally obliged. The applicability of these strategies is dependent on the mode of compliance the buyer is seeking. From the perspective of voters, benefits can be received in the form of payment, gift or wage, with different explicit and implicit meanings in terms of modes of compliance.

From these papers it is clear that what exactly constitutes acceptable influence and what does not, depends on the type of elections, the society in which the elections are being held and the participants of the elections. In the end, the matter is ultimately a subjective one. However, by determining the various characteristics of vote buying, and their respective ranges, it is possible to establish a pre-election consensus on allowed and disallowed practices. Such a pre-election consensus enables putting precise requirements on voting systems to support the one type of behaviour, while preventing the other type.

1.2 Outline of the Paper

In Section 2, we identify and categorise the types of attack that can be a threat to anonymity of the voter, including different types of vote buying and coercion. This analysis leads to a categorisation of anonymity in voting in terms of a) the strength of the anonymity achieved and b) the extent of interaction between voter and attacker, which is presented in Section 3. In Section 4, some of the combinations, including weak and strong receipt-freeness, are formalised in epistemic logic. The last section presents conclusions and future work.

2 Characteristics of Voter Influencing

In this section, we investigate the characteristics of voter influencing. The examples below are used as supporting guidelines throughout the section. These examples are deliberately without context – in lieu of what was established in Section 1.1. The reason for this is that the aim is to discover the generic characteristics involved, irrespective of social and electoral context. The examples are not meant to capture any precise attempt at influencing voters, but rather they convey a broad idea of a, possibly controversial, attempt at changing the outcome of an election by targeting the voters.

Example 1 (handout). At the polling station, I give each voter 100 euros together with mentioning my candidacy.

Example 2 (theme park). The district with the highest percentage of votes for me gets a theme park as my first act as elected official.

Example 3 (zalmstip). If I get elected, everyone gets 100 euros tax refund.

Example 4 (election promise). If I get elected, disabled child prodigies get 100 euros (i.e. children with a physical handicap, who are members of Mensa).

Example 5 (rf). I give 100 euro for anyone voting for the Democratic Party.

Example 6 (non-rf). I give 100 euro for anyone not voting for the Democratic Party.

Example 7 (reimburse). I provide a reimbursement for voters for time away from work.

The *zalmsnip* example is based on a tax rebate that occurred in the Netherlands in 1998 and 1999. The *rf* and *non-rf* examples are inspired by the notion of receipt-freeness. The *reimburse* example is inspired by this practice occurring in the 19th century in the Netherlands.

Note that – as long as there is no request to vote in a specific way – example 1 can be considered legitimate. Examples 3 and 4 are fabrications resembling possible election promises. Example 2 is dubious; examples 5–7 are outright illegal.

2.1 Legal and Illegal Influencing

Influencing voters can be done either legally or illegally. To avoid a legal discussion on what is allowed by which laws, we only focus upon characterising what is desirable. As established in the introduction, this is a subjective notion. The aim here is to outline the range of possibilities available, indicate where the boundary between desirable and undesirable lies and give a supportive reasoning for where we feel this boundary lies.

Note that, in general, there are two methods to influence a voter's vote:

coercion where voters are threatened to ensure compliance;

enticement where voters are seduced into compliance.

Whereas persuasion is allowed, buying and coercion are not. Both buying and coercion require proof of compliance, whereas persuasion does not. Both buying and persuasion are dependent on voluntary cooperation of the voter, coercion is not.

Voter influencing can be considered acceptable or unacceptable. What is considered acceptable depends on culture and the nature of the elections. That there can exist both acceptable and unacceptable variants of the above two methods is illustrated by the following list.

- *acceptable coercion* claiming that all other candidates have significantly worse plans for the voter
- *unacceptable coercion* threatening with physical violence in case of non-compliance
- *acceptable enticement* promising to lower taxes
- *unacceptable enticement* paying a voter to vote for you

The above list clearly indicates, that there is a distinction between acceptable influence and unacceptable influence. To establish the characteristics that together determine the acceptability, we construct an objective tree of voter influencing

in Section 2.2. Objective trees are attack trees (see [19,13]), but focus upon meeting goals instead of achieving attacks.

Our objective tree deviates slightly from normal attack trees. The purpose of our tree is to determine characteristics that distinguish acceptable from unacceptable influence. To elucidate these detailed characteristics, details need to be explicit in the tree. Hence, where normally attributes would be used, we promote these characteristics to leaves. This makes these characteristics explicit.

2.2 Classifying Vote Buying

Based on the literature, the examples and the analysis above, the tree in Figure 1 was constructed and dimensions of vote buying were clarified. The main goal in the tree is to buy a vote, by means of persuasion. The tree is thus from the perspective of a vote buyer. Where necessary, the range of possible values has been indicated in the tree (as leaves).

The tree is to be read as follows: before each entry, the type of the entry is marked ('or' for or-nodes, 'and' for and-nodes, 'leaf' for leaves). Or-nodes are

- **or** reward time
 - **leaf** before vote casting
 - **and** later
 - * **or** trust required
 - **leaf** rewarding sureness
 - **leaf** consequences of non-reward
 - **leaf** ensurance of compliance
 - * **or** hand out reward
 - **leaf** after vote casting
 - **leaf** after ballot box closes
 - **leaf** after vote counting
- **or** type of reward
 - **leaf** money
 - **leaf** goods
 - **leaf** immaterial
- **or** rewarding conditions
 - **leaf** cast vote
 - **leaf** election win
 - **leaf** unconditional
 - **leaf** complex
- **leaf** group size of reward receivers
- **or** proving compliance
 - **leaf** before rewarding
 - **leaf** after rewarding
 - **leaf** not required
- **leaf** reward related to election

Fig. 1. Objective tree for vote buying

nodes of which at least one of the branches must be satisfied, for an and-node to be satisfied, all branches must be satisfied.

In the tree, the characteristic *reward related to election* poses the question whether or not the reward can only be handed out by the election winner. An example of such rewards is granting amnesty, which is not possible unless elected to office.

As not all aspects of vote influencing are of direct interest to the buyer (e.g. how sure is delivery of the reward), other trees for different objectives have been constructed. Noting that this objective tree focuses upon vote acquisition only via rewarding leads to the following dimension:

type of compliance how is compliance achieved?

instrumental (by rewarding), normative (by convincing), coercive.

A vote buyer, however, is not interested in acquiring one vote, but in acquiring many votes. Specifically, a vote buyer's goal is to acquire enough votes to make a difference. This goal of a vote buyer encompasses the above objective of acquiring one vote. A vote buyer could use various means to attempt vote acquisition of many votes (as indicated by the examples). An analysis of this, similar to the above one, uncovered the following dimensions of approaches to vote buying:

non-compliance what impact does non-compliance have?

no rewarding, rewarded anyway.

focusability can only the relevant people be the sole targets of the vote buying method?

highly targetable, less targetable.

scalability how easy is it to employ this rewarding on a large scale?

high, medium, low.

costs do costs vary with the number of acquired votes or voters convinced?

variable, more fixed.

openness/publicity is the persuasion attempt general knowledge?

secret (known to buyer and seller(s)), not hidden, general knowledge.

Considering the point of view of the vote seller introduced the following dimensions for vote buying:

rewarding certainty can rewarding be avoided?

unavoidable, avoidable.

consequences of non-reward what impact will not rewarding have?

high impact, low impact.

Note that these two dimensions are closely related; they can be combined as **commitment to reward**.

Considering these various objectives together (i.e. vote seller, vote buyer for one vote and for a group of votes) led to the following dimension:

proof of compliance who should prove compliance?

proof by buyer, proof by seller. in case of vote buying, proof by seller is expected; in case of promises, proof by buyer is expected

One remarkable observation, given these dimensions, is that absence of receipts (receipt-freeness, see e.g. [2]) is not sufficient to prevent vote buying – it only suffices to prevent proving compliance. Forms of vote buying exist that do not require proving compliance by the individual voter, as can be seen in example 2.

2.3 Classifying the Targets

The set of possible targets for voter influencing can be characterised on various levels. From large to small, we distinguish the following:

- population* (1)
- \supseteq *eligible voters* (2)
- \supseteq *registered voters* (3)
- \supseteq *voters casting votes* (4)
- \supseteq *voters casting valid votes* (5)
- \supseteq *voters casting valid votes for vote buyer* (6)

Additionally, preferences with respect to elections and vote buying differ from person to person. Perhaps some individuals do not mind selling their votes, while others may find the practice so repugnant they will not vote for anyone involved with the practice – even if it is their preferred candidate. We distinguish the following dimensions for voters:

- will accept reward** yes / no
- initial preference** buyer’s choice / other
- awareness of attempt** none / heard rumours / fully aware
- is a desired target** yes / no
- cast vote** buyer’s choice / other

This classification can be applied to each of the sets 1–6. This classification extends the work of Acker [1], who classified voters targeted by election promises as follows:

- A.** Already compliant voters — these would have voted for the coercer without the election promise
- B.** Voters who change their votes — these vote for the coercer due to the election promise
- C.** Non-compliant voters — these do not vote for the coercer, despite the election promise

One category missing in that classification is explicitly included by our new classification: the set of voters who, as a result of the vote buying attempt, change their vote from “buyer’s choice” to “other”. Intuitively, these voters can be characterised as the voters who find vote buying so repugnant, that they will not vote for anyone involved with the practice.

2.4 Classification of the Examples

Below we classify our examples of voter influencing, according to the dimensions² established above. In addition, for each example we note which subset of voters is targeted.

Example 1 (handout). **conditions:** unconditional; **group size:** individual; **related:** no; **type:** instrumental; **non-compliance:** rewarded; **focus:** not targetable; **scalable:** low; **costs:** number of attempts; **publicity:** not hidden; **commitment:** unavoidable, low; **proof:** none.

Targetted at voters casting votes (class 4).

Example 2 (theme park). **conditions:** conditional; **group size:** collective; **related:** yes; **type:** instrumental; **non-compliance:** rewarded; **focus:** not targetable; **scalable:** high; **costs:** fixed; **publicity:** public; **commitment:** avoidable, high; **proof:** none.

Targetted at registered voters (class 3).

Example 3 (zalmstip). **conditions:** conditional; **group size:** collective; **related:** yes; **type:** instrumental; **non-compliance:** rewarded; **focus:** not targetable; **scalable:** high; **costs:** fixed; **publicity:** public; **commitment:** avoidable, high; **proof:** none.

Targetted at registered voters (class 3).

Example 4 (election promise). **conditions:** conditional; **group size:** collective; **related:** yes; **type:** instrumental; **non-compliance:** rewarded; **focus:** highly targetable; **scalable:** high; **costs:** fixed; **publicity:** public; **commitment:** avoidable, high; **proof:** none.

Targetted at a subgroup of registered voters (class 3).

Example 5 (rf). **conditions:** conditional; **group size:** individual; **related:** no; **type:** instrumental; **non-compliance:** unrewarded; **focus:** highly targetable; **scalable:** low; **costs:** number of acquired vote; **publicity:** not hidden; **commitment:** avoidable, low; **proof:** by voter.

Targetted at voters casting valid votes (class 5).

Example 6 (non-rf). **conditions:** conditional; **group size:** individual; **related:** no; **type:** instrumental; **non-compliance:** unrewarded; **focus:** highly targetable; **scalable:** low; **costs:** number of compliant voters; **publicity:** not hidden; **commitment:** avoidable, low; **proof:** by seller.

Targetted at voters casting valid votes (class 5).

Example 7 (reimburse). **conditions:** unconditional; **group size:** individual; **related:** no; **type:** instrumental; **non-compliance:** rewarded; **focus:** not targetable; **scalable:** medium; **costs:** number of requesting voters; **publicity:** public; **commitment:** unavoidable, -; **proof:** not required.

Targetted at registered voters (class 3).

² Both *time of reward* and *type of reward* have been left out, as these are already explicit in the examples.

We find that, based on our distinctions, we can easily classify these examples. The question remains which attributes indicate acceptable and unacceptable forms, respectively. From our examples and their intuitive acceptability, we propose that benefits that are related to the contested position, are unconditional and openly announced, are most likely to be found legitimate.

2.5 Conclusions on Vote Influencing

We conclude that vote buying involves much more than offering money in exchange for a proof of compliance. Attributes that make it likely for an action to be considered vote buying include:

- unrelated to contested position;
- reward independent of being elected;
- reward conditional on compliance (therefore proof by seller);
- highly targetable;
- variable costs (related to individual payment);
- secrecy of the attempt.

Individuality does *not* make things worse; buying a whole district is in itself no better than buying votes one by one. The publication of any election result on a level lower than strictly necessary (e.g. per polling station) facilitates collective vote buying, and would best be eliminated from this perspective. Electronic voting can facilitate such a transition, by storing votes independently from the place where they were cast.

However, this is by itself not sufficient to stop collective vote buying. If a buyer wants to buy a set of votes, and knows with 70% certainty that an individual voter complies, she can be fairly sure that if she buys a large amount of votes, 70% of the votes will be hers, due to the law of large numbers.

Conversely, if in a particular set of votes 70% is for the buyer, she can derive that a voter whose vote is in this set has voted for her with a 70% probability. This particular observation gives rise to the notion of *probabilistic vote buying* – where a buyer requires not exact votes, but is satisfied by a (significant) change in the distribution of votes.

Furthermore, the *non-rf* example showed that it is possible to *discourage* voting for a certain party or candidate in an action of vote buying. This is an example of voter influencing that cannot be directly described in terms of the intuitive notion of receipt-freeness, but is close to it.

In the next section, we categorise anonymity in voting based on these observations.

3 Dimensions of Anonymity in Voting

Traditionally, the concept of vote buying has been related to the possibility of providing a proof of one's choice. The notion of receipt-freeness was proposed to prevent such a proof. However, our framework developed in the previous sections shows that a proof is not always necessary. The following actions would be possible without a proof of the voter's choice in a strict sense:

- rewarding the voter if she does *not* vote for a specific party or candidate (related to negative proof);
- rewarding the voter if it is *likely* that she made a certain choice.

It can be enough for a buyer to hand out the reward if a voter can show that she did *not* vote for two of the buyer’s opponents (example 6). The buyer could also pay a voter if after observing the outcome, it is *more* likely that this voter voted for him than that another voter voted for him. If this can be observed from messages sent in the voting protocol, this should be addressed by computer science verification methods. One could also derive this from voter behaviour [4], but that is hard to prevent using computer science tools.

Each of the notions of privacy, receipt-freeness and coercion-resistance can be investigated with respect to these scenarios:

	weak	strong	probabilistic
privacy	this work	this work	Barghava et al.
receipt-freeness	this work	this work	–
coercion-resistance	Delaune et al.	–	–

The distinction between the three notions of anonymity depends on the relation between voter and attacker. In case of privacy, no cooperation of the voter is assumed; the attacker tries to find out about the voter’s choice without cooperation the voter. In case of receipt-freeness, the voter cooperates by sending information to the attacker. In case of coercion-resistance, the voter even accepts instructions from the attacker.

The *weak* variant of the notion then concerns the situation that an attacker cannot be sure of the voter’s choice. The *strong* variant of the notion concerns the situation that an attacker cannot be sure of what the voter did *not* choose either. The *probabilistic* variant of the notion concerns the situation what an attacker cannot deduce anything from the *probability distribution* of a voter’s choices.

In the next section, we formalise the notions of strong and weak privacy and receipt-freeness in an epistemic framework. Probabilistic privacy has been investigated in [3] (where it is called probabilistic anonymity). The notion of coercion-resistance is defined in [6]. Filling the remaining fields in the table based on these definitions is future work.

4 Formalising Anonymity Properties

Garcia, Hasuo, Pieters and Van Rossum defined a framework for describing anonymity properties of protocols in epistemic logic [7]. We use this framework as the basis for our definitions of privacy and receipt-freeness. Their definitions are based on a formalisation of runs in protocols. For the formal definitions, we refer to the original paper.

Based on observational equivalence of runs, the following notions are defined formally in the original paper. We include the informal definitions here. The

formula A Sends m to B means: at some stage in the run, A sends a message to B which contains m as a subterm. A Sends m means that A sends the message m to someone. The formula A Possesses m means: after the run has finished, A is capable of constructing the message m . The formula A Originates m means that A Sends m , but A is not relaying. More precisely, m does not appear as a subterm of a message which A has received before.

The formula $\Box A\varphi$ is read as “after the run is completed, the agent A knows that φ is true”. The formula $\Diamond A\varphi$ is short for $\neg\Box A\neg\varphi$ and read as “after the run is completed, the agent A suspects that φ is true”.

Garcia et al. define the information hiding properties of sender anonymity, unlinkability and plausible deniability using the notion of an *anonymity set* (a collection of agents among which a given agent is not identifiable) in epistemic logic as follows:

Definition 1. (*Sender anonymity*) Suppose that r is a run of a protocol in which an agent B receives a message m . We say that r provides sender anonymity with anonymity set AS if it satisfies

$$r \models \bigwedge_{X \in AS} \Diamond B(X \text{ Originates } m).$$

This means that, as far as B is concerned, every agent in the anonymity set could have sent the message.

Definition 2. (*Unlinkability*) A run r provides unlinkability for users A and B with anonymity set AS iff

$$r \models (\neg\Box\text{spy}\varphi(A, B)) \wedge \bigwedge_{X \in AS} \Diamond\text{spy}\varphi(X, B),$$

where $\varphi(X, Y) = \exists n. (X \text{ Sends } n \wedge Y \text{ Possesses } n)$.

Intuitively, the left side of the conjunction means that the spy (the adversary) is not certain that A sent something to B . The right conjunct means that, according to the spy, every other user could have sent a message to B . Similarly, unlinkability between a user A and a message m could be defined as $\models \neg\Box\text{spy}(A \text{ Sends } m) \wedge \bigwedge_{X \in AS} \Diamond\text{spy}(X \text{ Sends } m)$.

In certain circumstances (e.g. relays), agents might be interested in showing that they did not know that they had some sensitive information m . This might be modeled by the following epistemic formula:

Definition 3. (*Plausible deniability*) Agent A can plausibly deny message m in run r iff

$$r \models \Box\text{spy}\neg(\Box A(A \text{ Possesses } m)).$$

This formula is read as: the spy knows that A does not know that she possesses m .

We extend this set of definitions by providing the additional property of *receipt-freeness*. Receipt-freeness of an agent A with respect to a message m (e.g. a vote) intuitively means that A cannot send a message m' to the spy that proves that she sent m in the past. For this purpose, the definition of plausible deniability is too strong, since A *does* know that she possesses m . Sender anonymity is particularly useful for providing anonymity of the voter with respect to the election authorities, but in receipt-freeness, A herself tries to communicate with the spy. Instead, it should not be possible to link A to her vote. Thus, unlinkability seems the most natural property to base our definition of receipt-freeness upon.

In the anonymity framework, the concept of anonymity set is used to define the set of entities between which an observer should not be able to distinguish. To apply the framework to votes, we need to adapt the concept of anonymity set. In voting, we are sure that each (actual) voter submits a vote. Therefore, the point is not whether any other user in an anonymity set could have sent the message, but *whether the voter could have submitted any other vote*. Therefore, we define an anonymity set of *messages*, AMS, instead of an anonymity set of agents. This set typically consists of all possible votes.

First of all, we define the notion of (weak) privacy. This can be achieved without referring to the anonymity set.

Definition 4. (*Weak privacy*) *A run of a protocol is weakly private for agent A with respect to message m iff*

$$r \models \neg \Box \text{spy}(A \text{ Sends } m)$$

To be able to define receipt-freeness, we need to have a way to extend a given run with one message: the receipt. We write this as $r.(A \rightarrow B : m)$ for a given run r , message m (the receipt), sender A and receiver B . For A to be able to send the receipt, she needs to have the message in her possessions at the end of the original run. The new run does *not* need to be a run of the protocol. It *does* need to be legitimate with respect to the initial possession function. $\text{Poss}_{\text{IP}_0}(r, A, |r| - 1)$ denotes the possessions of agent A at the end of the original run (see [7]).

Definition 5. (*Weak receipt-freeness*) *A run of a protocol is weakly receipt-free for agent A with respect to message m iff for all $m' \in \text{Poss}_{\text{IP}_0}(r, A, |r| - 1)$,*

$$r.(A \rightarrow \text{spy} : m') \models \neg \Box \text{spy}(A \text{ Sends } m)$$

Weak receipt-freeness implies that the voter cannot prove to the spy that she sent message m during the protocol, where m is the (part of a) message representing the vote. However, this notion is still fairly limited. For example, suppose that the spy wants the voter to vote for party X . Suppose, furthermore, that the voter instead chooses to vote Y , which is represented by message m in the above definition. Now, if the voter cannot show that she voted Y , this protocol is receipt-free with respect to the definition above. However, if the spy can acquire information which proves that the voter did *not* vote X , the spy will not be satisfied. Therefore, we introduce stronger notions of privacy and receipt-freeness as well.

Definition 6. (*Strong privacy*) A run of a protocol is strongly private for agent A with respect to a message m in anonymity set AMS iff

$$r \models (\neg \Box \text{spy}(A \text{ Sends } m)) \wedge \bigwedge_{m'' \in \text{AMS}} \Diamond \text{spy}(A \text{ Sends } m'')$$

Definition 7. (*Strong receipt-freeness*) A run of a protocol is strongly receipt-free for agent A with respect to a message m in anonymity set AMS iff for all $m' \in \text{Poss}_{\text{IP}_0}(r, A, |r| - 1)$,

$$r.(A \rightarrow \text{spy} : m') \models (\neg \Box \text{spy}(A \text{ Sends } m)) \wedge \bigwedge_{m'' \in \text{AMS}} \Diamond \text{spy}(A \text{ Sends } m'')$$

Here, no matter what information the voter supplies to the spy, *any* vote in the anonymity set is still possible. This is represented by the “suspects” symbol $\Diamond \text{spy}$. In other words, for all possible votes, the spy still suspects that the voter cast this particular vote; or: the spy is not certain she did *not* cast this vote. This requires that at least one message has been received (i.e. at least one vote has been cast) for every message (vote) $m'' \in \text{AMS}$. Otherwise, the spy could observe from the results that no-one, in particular not voter A , cast a certain vote. Thus, for votes, AMS is a subset of the set of candidates who received votes.

Notice that this definition is analogous to the definition of unlinkability of Garcia et al.

Theorem 1. *If a run of a protocol is strongly receipt-free for agent A with respect to message m in anonymity set AMS , then it is also weakly receipt-free for agent A with respect to message m .*

Proof. This follows directly from the definitions.

In the framework, it can be defined which part of the messages the spy can observe. If the spy could read all the messages, the voter only needs to supply the secret keys in order to provide a receipt. This is not what is commonly understood by analyzing receipt-freeness. Instead, there are certain messages in the protocol that the spy is not assumed to have access to (when the voter is in a “voting booth”).

In our definition, we deviate from the approach by Delaune et al. [6]. Intuitively, receipt-freeness is achieved if a voter does not possess convincing, exclusive evidence of how she voted. The approach by Delaune et al. defines receipt-freeness using two voters (to preserve indistinguishability of the result). By focusing on the actual receipt, our definition only relies on one voter, and thus remains closer to the intuition. The indistinguishability is made explicit in our definition by AMS , which does not need to encompass all candidates but can be confined e.g. to all candidates for whom at least one vote was cast.

5 Conclusions and Future Work

In this paper, we examined various dimensions of the notion of anonymity in electronic voting. Based on an analysis of the acceptability of vote buying and coercion, we found that many different dimensions contribute to whether an action is classified as vote buying or coercion, or not. Having established these dimensions, we distinguished between *weak*, *strong* and *probabilistic* notions of anonymity in voting. This distinction applies to privacy, receipt-freeness and coercion-resistance.

Following the definitions of anonymity in [7], we introduced an approach to formally verify weak and strong receipt-freeness in epistemic logic. To the best of our knowledge, we are the first to do so. The approach has not been tested on real protocols thus far.

One of the main benefits of our approach is the intuitive definition that it provides for receipt-freeness. As opposed to other approaches, especially [6], the “receipt” can easily be distinguished in our model as a separate message that the voter sends to the spy. Instead of investigating whether the spy can recover the vote from forwarded messages, we judge whether the spy really *knows* what the voter’s choice was, based on any possible receipt. This notion of *knows* is characteristic for the epistemic logic approach, and this justifies our choice for the anonymity framework of [7] as a basis.

While our approach captures receipt-freeness, the investigation of the notion of vote influencing clearly indicates that compliance with a technical criterion such as receipt-freeness is insufficient to prevent voter influencing – if unaware, a voter could still be susceptible to voter influencing. Further study in this area (see also [16]) is needed to ensure that security requirements not only enable secure voting, but that voters are sufficiently aware of the security they provide.

In future work, we aim at providing an alternative definition of receipt-freeness in our model, based on the knowledge of the spy instead of on extension of a run. We hope to prove that the two definitions are equivalent. Moreover, we wish to apply the definitions to existing voting protocols, in order to prove (or disprove) receipt-freeness. It may also be interesting to investigate the relation between verifiability [15] and receipt-freeness in epistemic logic, since both receipt-freeness and verifiability are based on an agent’s knowledge instead of its possessions.

As we have seen, voter influencing can take many forms, and only some can be hindered by technological means. However, being more precise in what we can and cannot achieve in technology can lead to better decisions on which protocols are acceptable and which are not. Again, the latter will depend on what is seen as acceptable in a particular culture, which has been determined during a long history of success and fraud.

References

1. van Acker, B.: Remote e-voting and coercion: a risk-assessment model and solutions. In: Electronic Voting in Europe 2004, pp. 53–62 (2004)
2. Benaloh, J.C., Tuinstra, D.: Receipt-free secret ballot elections (extended abstract). In: Proc. 26th ACM Symposium on the Theory of Computing (STOC), pp. 544–553. ACM, New York (1994)

3. Bhargava, M., Palamidessi, C.: Probabilistic anonymity. In: Abadi, M., de Alfaro, L. (eds.) CONCUR 2005. LNCS, vol. 3653, pp. 171–185. Springer, Heidelberg (2005)
4. Brusco, V., Nazareno, M., Stokes, S.C.: Vote buying in Argentina. *Latin American Research Review* 39(2), 66–88 (2004)
5. Delaune, S., Kremer, S., Ryan, M.D.: Receipt-freeness: Formal definition and fault attacks (extended abstract). In: Proceedings of the Workshop Frontiers in Electronic Elections (FEE 2005), Milan, Italy (September 2005)
6. Delaune, S., Kremer, S., Ryan, M.D.: Coercion-resistance and receipt-freeness in electronic voting. In: Proceedings of the 19th IEEE Computer Security Foundations Workshop (CSFW 2006), Venice, Italy, July 2006. IEEE Computer Society Press, Los Alamitos (2006)
7. Garcia, F.D., Hasuo, I., Pieters, W., van Rossum, P.: Provable anonymity. In: Küsters, R., Mitchell, J. (eds.) 3rd ACM Workshop on Formal Methods in Security Engineering (FMSE 2005), pp. 63–72. ACM Press, New York (2005)
8. Hasen, R.L.: Vote buying. *California Law Review* 88(5), 1323–1371 (2000)
9. Hirt, M., Sako, K.: Efficient receipt-free voting based on homomorphic encryption. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 539–556. Springer, Heidelberg (2000)
10. Jonker, H.L., de Vink, E.P.: Formalising receipt-freeness. In: Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC 2006. LNCS, vol. 4176, pp. 476–488. Springer, Heidelberg (2006)
11. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: Proc. WPES 2005. ACM, New York (2005)
12. Kochin, M.S., Kochin, L.A.: When is buying votes wrong? *Public Choice* 97, 645–662 (1998)
13. Mauw, S., Oostdijk, M.: Foundations of attack trees. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 186–198. Springer, Heidelberg (2006)
14. Mauw, S., Verschuren, J.H.S., de Vink, E.P.: A formalization of anonymity and onion routing. In: Samarati, P., Ryan, P.Y.A., Gollmann, D., Molva, R. (eds.) ESORICS 2004. LNCS, vol. 3193, pp. 109–124. Springer, Heidelberg (2004)
15. Pieters, W.: What proof do we prefer? variants of verifiability in voting. In: Ryan, P., Anderson, S., Storer, T., Duncan, I., Bryans, J. (eds.) Workshop on e-Voting and e-Government in the UK, Edinburgh, February 27–28, pp. 33–39. e-Science Institute, University of St. Andrews (2006)
16. Pieters, W.: La Volonté Machinale – understanding the electronic voting controversy. PhD thesis, Radboud University, Nijmegen, The Netherlands (2007)
17. Sako, K., Kilian, J.: Receipt-free mix-type voting scheme – a practical solution to the implementation of a voting booth. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 393–403. Springer, Heidelberg (1995)
18. Schaffer, F.C., Schedler, A.: What is vote buying? In: Schaffer, F.C. (ed.) *Elections for Sale: The Causes and Consequences of Vote Buying*. Lynne Rienner, Boulder CO (2007)
19. Schneier, B.: Attack trees: Modeling security threats. *Dr. Dobb's journal* (December 1999)

Anonymous One-Time Broadcast Using Non-interactive Dining Cryptographer Nets with Applications to Voting

Jeroen van de Graaf

Departamento de Computação
Universidade Federal de Ouro Preto
35400-000 Ouro Preto (MG), Brazil

Abstract. All voting protocols proposed so far, with the exception of a few, have the property that the privacy of the ballot is only computational. In this paper we outline a new and conceptually simple approach allowing us to construct a protocol in which the privacy of the ballot is unconditional. Our basic idea is to modify the protocol of Fujioka, Okamoto and Ohta [1], which uses blind signatures so that the voter can obtain a valid ballot. However, instead of using a MIX net, we use a new broadcast protocol for anonymously publishing the vote, a Non-Interactive variation of the Dining Cryptographer Net.

1 Introduction

1.1 Motivation

Voting protocols are often divided in three categories: based on MIX networks, based on blind signatures, and based on homomorphic encryption. See for instance [2] for a somewhat simplified but nevertheless interesting overview. A major flaw of most voting protocols proposed so far, is that the privacy of the ballot is only computational:

MIX. For protocols based on MIX nets [3] this is so since, besides the usual assumptions about the authorities not revealing the permutations, their security is also based on RSA or ElGamal: if one knows the factor of the moduli used for mixing, one can trace back a vote through the cascade of mixes and find out who submitted the vote.

Blind signatures. The blind signature [4] allows a voter to obtain a signed ballot from a voting authority, who will cross out the voter from the list of voters. After unblinding his vote, the voter needs to publish his ballot anonymously. All protocols published so far use a MIX net for this purpose, which reduces it to the previous case.

Homomorphic encryption. A homomorphic encryption scheme (see for instance [5]) uses a clever way to encrypt each individual vote, such that by manipulating (in most cases: multiplying) the encryptions the votes can be tallied, so that the problem reduces to decrypting collectively some specific value. It is obvious that someone with infinite computational power could decrypt all the ballots and therefore who voted for whom.

This flaw is really worrisome for the following reason: with storage becoming cheaper and cheaper every year, we must assume that all data that has been made public through

an election protocol, will never be erased, i.e. that some copy of it will be stored forever. We also must assume that at some point in the future it will be possible to break the underlying computational assumption, and then it will become public who voted for whom. Though one can argue that this information might have become irrelevant after many decades, this point is more important than it seems. For instance, people might like to know who the President of the United States voted for when he was young. He might have had a flirt with the communist party, who knows. Even today historians will find it interesting to know Churchill's voting behavior in 1900, when he was about 25 years old.

Real world voting systems always have had the property that the vote (the information containing the voter's choice) is irretrievably destroyed. Newly proposed protocols should have this property too. **Computational privacy is not enough for voting**, since one day or another the computational assumption will be broken. Of course, in less important elections it might be acceptable that the privacy of the vote is only computational. But it should be pointed out that estimating for how long the privacy of the ballot will be preserved is notoriously difficult, as various examples exist of computational assumptions being broken much earlier than expected.

1.2 Outline of This Paper

In this paper we describe a conceptually simple voting protocol which has unconditional voter privacy. The main ingredient is that we propose a non-interactive (i.e. one round) version of the well-known Dining Cryptographers protocol [6], which, as far as we know, has not been published before, and is of separate interest. Though the basic idea is simple, some technical subtleties need to be resolved since, unlike the original protocol, there is only one round. In the next section we will give a high level sketch of the proposed voting protocol, whereas Section 3 contains a detailed description of the non-interactive variation of the Dining Cryptographers protocol.

1.3 Relation to Other Work

We are not aware of any paper that proposes the non-interactive use of the Dining Cryptographers protocols. Bos [7] presents a voting protocol based on DC nets, but this protocol makes the (in our opinion unrealistic) assumption that all voters are simultaneously on-line. The protocol of [8] uses techniques whose mathematics is similar to the math of DC nets. This protocol is exclusively devoted to voting and uses only one slot (as is the case with [7]), while we propose the use of many slots to allow each participant to broadcast his vote.

As a consequence, their protocol has a much smaller message size ours but is also less general. It only deals with Yes/No votes, but if the number of candidates or the way preferences are expressed changes, the protocol has to be re-engineered. This is not the case for our protocol; we basically propose an anonymous broadcast channel which is insensitive to the exact lay-out of the message sent through it.

Another interesting paper is [9], which argues for "everlasting privacy" in voting, like we do. It uses a non-interactive bit commitment scheme as the underlying assumption, a primitive that we also need, but in most other aspects the techniques used in their paper are completely different.

2 A High-Level Sketch of the New Voting Protocol

The basic idea for this new voting protocol consists of two main ingredients, which are presented separately. Note that the first ingredient is not new.

2.1 The First Ingredient: Blind Signatures

The idea is that by using a Chaum-like blind signature, the voter gets a valid ballot from the voting authority. In particular there is the Fujioka, Okamoto e Ohta protocol [11] that could be used here. This protocol allows the voter Alice to contact a Ballot Issuing Authority in order to submit a blinded ballot. The Authority responds by (blindly) signing Alice's ballot and sending it back to her. Because the blinding process is perfect (since all blinding factors are equiprobable, all possible votes are), the authority obtains no information whatsoever about Alice's vote.

It is important that the Authority marks Alice's name on the list of eligible voters, thus avoiding that Alice tries to vote twice. Equally important is that both parties sign their messages and maintain records of them, in order to resolve later disputes.

When receiving the message from the Authority, Alice unblinds it and verifies that it contains a valid, signed ballot.

2.2 Mixnets and Their Disadvantages

The next step of the Fujioka, Okamoto e Ohta protocol is that Alice must cast her ballot through some anonymous broadcast mechanism. One possible way to accomplish this is by using MIX-nets. This allows the voter to submit his ballot in encrypted form, usually with several layers of encryption. All the ballots related to one election could be placed on a web site, for instance. Now Mixing Authorities are involved in decrypting the batch of ballots layer by layer, while shuffling the intermediate results. If the voter followed the protocol correctly, the last decryption will show the vote, signed by the Ballot Issuing Authority, in the clear. Later some audit protocol is run to verify (with high probability) that none of the mixing authorities cheated.

When sending her vote, there is no reason for a voter to withhold her identity. On the contrary, one can imagine that the identity of the voter is placed next to his encrypted vote, since the privacy is supposedly guaranteed by the mix net. Consequently, when (and not: if) at some moment in the future the private (RSA or ElGamal) keys are broken, the permutation used by each mix can be reconstructed and all the links can be established between the encrypted vote as submitted by the user, and the fully decrypted result of the mix. So the privacy of the ballot will be violated.

2.3 The Second Ingredient: Non-interactive Dining Cryptographers

This is the reason we introduce a second and new ingredient of our protocol: instead of using a (Chaum) mix-net, the voter uses a non-interactive (Chaum) Dining Cryptographers network to post her vote. This protocol can be informally described as follows:

Three cryptographers are having dinner. When they have finished their meal, the waiter informs them that their meal has been paid already. The cryptographers decide

they want to find out whether the meal was paid by an outsider (the NSA, say), or by one of the three present. However, if the payer is one of them, the identity of this payer should remain secret, i.e. the payer should be anonymous. In order to accomplish this, they decide to run the following protocol:

1. Each one flips a coin, and shares the result with his neighbor on the right.
2. Each one looks at his own coin and the one of his left neighbor. The two coins can have the same face up (heads-heads, tails-tails) or faces up (heads-tails, tails-heads). Each person announces publicly SAME or DIFFERENT, but with the additional rule that the person who paid reverses his statement (he “lies”).
3. When an odd number of persons announces DIFFERENT they know one of them has paid; when an even number of persons announces DIFFERENT they know an outsider paid.

It is not difficult to see how this protocol extends to many bits and many parties. See [6] and [7], Chapter 2. However, the model used in these papers is completely based on a network setting: people connected to each other through a local area network (Ethernet at the time), broadcasting messages to each other. An implication of this model is that if problems occur, new transmission rounds are available to resolve those problems.

Here we propose a non-interactive variation of the DC net, an idea which, as far as we know, has not been explored in the literature. The idea is that each voter submits **once** a bit sequence, properly identified, which gets published on the Bulletin Board. This sequence might be long but should still be reasonable for transmission, in the order of ten of megabytes, say. The non-interactive variation works very much like the original DC proposal:

1. in a preliminary phase, each pair of parties exchanges random bits.
2. based on these random bits and on the party’s input, it broadcasts a message.
3. all message are combined in such a way that all the random bits cancel, and only the inputs of all parties remain.

It is well known that DC-nets provide unconditional anonymity; this follows from the fact that the probability distribution on the random bits is uniform. See [6] for details.

In the non-interactive case, to avoid collisions, most bits of the sequence published will contain the xors of the bits exchanged with other parties, but no message bits. Only at a very short interval a message, containing the vote with its signature, will be added to the xors as well. Very short signatures that allow blinding can be obtained using elliptic curves, resulting in signature lengths of 160-200 bits.

Note that the signature scheme only need to be resistant to attacks from the voters for the duration of the election to avoid any insertion of false votes. If the signature scheme gets broken after the election, the result is not compromised; indeed, revealing the private key after the election has completed (more precisely: when no more ballots can be issued by the Ballot Issuing Authority) would not affect security.

After each participant has submitted (published) his sequence, the bitwise xor for each bit position over all the sequences is computed, yielding anonymously the net messages published by the participants, i.e. the votes. A tallying authority calculates the final result, which can be verified by any scrutineer who cares to.

2.4 Some Subtle Issues

Ideally this would be the end of it, but there are three problems to be resolved:

- A) collisions;
- B) disrupters;
- C) voters that don't show up (participated in the preliminary phase but did not submit a sequence).

Ad A: The collision probability must be kept very low since it would imply that two (random) votes get lost. However, there exist various straightforward techniques to keep the collision probability low. They basically consist of increasing the length of the bit sequence, or of sending the same message various times, through a different channel or through the same one. See Section 4.1 for discussion.

Ad B: Disrupters: this is truly an annoying problem because in the traditional (interactive) DC setting catching disrupters is already problematic: all participants have to engage in an on-line protocol to identify and drive out the disrupter(s). In our case we cannot catch disrupters afterwards, so we must catch them when they submit. This can be done by having the parties commit on their random bits exchanged. To preserve privacy we will need a Bit Commitment Scheme that is computationally binding and unconditionally hiding. Then we will use a cut-and-choose protocol in which the sender shows that he is following the protocol. In particular he needs to show that for the whole sequence, except the part that contains the message (vote), the bits are truly the result of xors of bits already committed to. Whether he follows the protocol for the bits dedicated to the message we could check but don't have to; he could send in garbage, but at least he won't be disrupting the channel.

Ad C: A no-show is someone who went through the initial phase of the protocol, has shared their random bits with other parties, but did not submit any sequence. Their identity will become known, and the best solution is to have people who have interacted with them recalculate their submissions. Alternatively, in a setting of a small set of authorities who exchange random bits with each voter, the authorities can simply disregard the random bits of the no-show(s).

Note that in a situation where there is a relatively large amount of trust between the participants, items B) and C) are of no concern.

3 A Detailed Description of the Non-interactive Anonymous Broadcast Protocol

In this section we describe a Non-Interactive version of the Dining Cryptographers protocol. Though the motivation for this protocol is voting, we describe the protocol in a general setting, i.e. we do not use voting-specific terminology. But we do assume a the same message size for all participants.

Some parts of the protocols are of a challenge-response nature, in which the responses are always either random bits or random permutations (which, of course, can be obtained from random bits). In the protocol description we will write that a party commits, and then receives a challenge from a trusted random source. However, it will

be understood that this is implemented using the Feige-Shamir heuristic: after having fixed the commitments, the party applies a hash function and uses the results for the challenge. This technique is generally believed to be secure, and avoids the interaction.

3.1 Notation

We suppose there are P participants \mathcal{P}_i , with $i \in \{1 \dots P\}$. The purpose of each participant \mathcal{P}_i is to publish a message v_i anonymously. To this end a DC channel is available of total size N bits, which is divided into S slots, each of size L . We define the net input of participant \mathcal{P}_i to the DC channel as $M_i = M_i[1] \dots M_i[S]$; here each $M_i[s]$ denotes a slot. Now \mathcal{P}_i may occupy at most 1 out of those S slots to publish v_i , so there is one $s \in \{1 \dots S\}$. The other $S - 1$ slots must remain empty (contain zeroes), i.e. for $s' \neq s$ we have that $M_i[s'] = 0^L$, a string consisting of L zeroes.

As in the original DC protocol, any pair of participants $\mathcal{P}_i, \mathcal{P}_j$ can choose to engage in an exchange of random bits. If this is the case they share an edge in the so-called privacy graph and we call them neighbors. The privacy properties of our protocol are identical to those described in the original paper.

We introduce the following notation:

- $M[s]$ is the s th slot of M
- $M[[u]]$ is the u th bit of M
- We denote the random string of size $N = SL$ shared between \mathcal{P}_i and \mathcal{P}_j with R_{ij} . If no sharing takes place, we define $R_{ij} = 0^N$.
- The overall random string used by \mathcal{P}_i for encryption is $R_i = \bigoplus R_{ij}$, where j ranges over the neighbors of \mathcal{P}_i .
- \mathcal{P}_i 's overall contribution to the channel is called C_i , and we have therefore that $C_i = M_i \oplus R_i$.

3.2 Bit Commitments with XOR

An essential property of our protocol is that each participant must commit to his contribution, and show that it has the proper format, though without showing the value. Since we want unconditional privacy, we obviously need a bit commitment scheme that is unconditionally hiding and computationally binding. There are various options here, but for concreteness we use a bit (string) commitment scheme based on hash functions, as presented in [10].

Actually, ordinary bit commitments are not good enough for our protocol, since we will need a way to prove that linear relations between bit commitments hold without opening the values. For instance, we would like to show that $\overline{x_1} \oplus \overline{x_2} \oplus \dots \oplus \overline{x_k} = 0$. That is, we want to show that the equality $x_1 \oplus x_2 \oplus \dots \oplus x_k = 0$ holds without revealing any other information about the x_i . We will show a general construction to accomplish this property for *any* kind of bit commitment, at the expense of a factor $2K$, where K is a security parameter.

The solution presented here, attributed to Bennett and Rudich, is described Section 2.2 of [11], where it is called "Bit Commitment with XOR", abbreviated BCX. The idea is to represent each BCX bit commitment as a vector of pairs of simple bit commitments, such that each pair xors to the committed bit value. This allows for challenges

on one half of the bit commitment, without revealing its value. We describe the scheme here informally, using a simple example. A more formal description is given in [11], which also shows that the scheme generalizes to proving linear relations between many bit commitments.

Using this approach, a bit commitment to $x = 1$ could for instance be represented by: $\vec{x} = \langle (\bar{0}, \bar{1}), (\bar{1}, \bar{0}), (\bar{0}, \bar{1}), (\bar{0}, \bar{1}), (\bar{1}, \bar{0}) \rangle$, where $K = 5$, an artificially low value. The two components (columns) are labeled 0 and 1, also named *left* and *right*; the rows are labeled from 1 to K . Suppose \mathcal{A} is committed to $x = 1$ and also to $y = 0$ as follows: $\vec{y} = \langle (\bar{1}, \bar{1}), (\bar{1}, \bar{1}), (\bar{0}, \bar{0}), (\bar{1}, \bar{1}), (\bar{0}, \bar{0}) \rangle$, and that she needs to prove that the two bit commitments are different, i.e. that $x \oplus y = 1$.

1. \mathcal{A} tells for each pair of \vec{x} and \vec{y} whether the left component is equal or different. Or, equivalently, she opens the values $z_k = x_{k0} \oplus y_{k0}$ for $k = 1, \dots, K$.
2. \mathcal{A} receives K challenge bits b_1, \dots, b_K from the trusted random source.
3. For each $b_k = 0$, \mathcal{A} is required to open the left component, x_{k0} and y_{k0} , of the k th pair of \vec{x} and \vec{y} , and \mathcal{B} must check that they are equal; if $b_k = 1$ then \mathcal{A} opens x_{k1} and y_{k1} and \mathcal{B} must check that they are different (their mod 2 sum adds to 1). This must be executed for each challenge bit b_1, \dots, b_K .

It is easy to see that \mathcal{A} does not reveal the actual values of the bit commitments through this protocol since only either the left or the right component of each pair is revealed, while the value is defined as the xor of both. As far as the binding property is concerned: obviously, for each row in which she tries to cheat, \mathcal{A} gets caught with a probability $1/2$.

It is important to observe that in the protocol of (in)equality between two BCXs, the unopened halves are not lost (useless), implying that the BCX can (and must) be preserved for future use. For instance, after a proof that $\vec{x} = \vec{y}$, the remaining halves can constitute a new BCX \vec{t} with the property that $t = x = y$. Note that the view of the protocol showing inequality has the effect of flipping the semantics of the corresponding bit commitment for the k th pair: $t = x_{kb'_k} \oplus y_{kb'_k} \oplus z_k$, where $b'_k = 1 \oplus b_k$, the complement of b_k .

3.3 Preliminary Phase

As in the original DC protocol, during the preliminary phase each pair of neighbors creates their random bit string R_{ij} of size N . But unlike the original protocol, we require that both \mathcal{P}_i and \mathcal{P}_j commit individually to each bit of R_{ij} using the BCX bit commitment scheme explained above. To avoid any type of collusion between them, it is essential that they show to the world (the other participants, and any other observer) that they are committed to the same value.

We do this as follows: we first consider \mathcal{P}_i and \mathcal{P}_j as one party, written \mathcal{P}_{ij} , who will jointly create a set of N BCXs, but of size $2K$ instead of K . (If they cannot agree on how to do this jointly, we assume that at least one party aborts and that this particular pair of parties will not contribute to the overall protocol.) They prove the well-formedness to the other participants by showing that all pairs of the same BCX \vec{x} encode the same values, i.e. that $x_{k0} \oplus x_{k1} = x$ for $k \in \{1, \dots, 2K\}$. This is done as follows:

1. \mathcal{P}_{ij} creates an additional BCX \vec{y} of the same value, i.e. $x = y$.
2. The trusted source of randomness supplies $2K$ challenge bits b_i , as well as a random permutation σ on $\{1, \dots, 2K\}$.
3. \mathcal{P}_{ij} proves equality between \vec{x} and \vec{y} , applying the permutation σ to shuffle the pairs, i.e. by showing that either $x_{k0} = y_{\sigma(k)0}$ or $x_{k1} = y_{\sigma(k)1}$, depending on the value b_k .

If \mathcal{P}_{ij} tries to cheat on a subset \mathbf{A} , this remains undetected only if the permutation σ maps \mathbf{A} onto itself. If $a = \#\mathbf{A} > 1$ this happens with probability $\binom{2K}{a}/(2K)!$. By repeating the protocol this probability can be reduced to any desired level of security.

After this protocol has completed, \mathcal{P}_i and \mathcal{P}_j split their double BCX of size $2K$ in two BCXs of size K by dividing the pairs evenly between them, for instance \mathcal{P}_i stays with the first K pairs $1, \dots, K$ and \mathcal{P}_j stays with the second K pairs $K + 1, \dots, 2K$.

3.4 Publication Phase

During the second phase of the protocol each participant decides which message v_i he wants to publish, for instance a signed vote.

This part consists of the following substeps:

1. \mathcal{P}_i commits to his input M_i , which contains v_i , and proves that it has the proper format;
2. \mathcal{P}_i commits to the contribution C_i and proves that it has the proper format.

Commitment and proof of M_i

1. Let v_i be the message that \mathcal{P}_i wants to publish. \mathcal{P}_i now creates M_i by selecting a slot $s \in \{1, \dots, S\}$ randomly. He sets $M_i[s] := v_i$, whereas for $s' \neq s$ he sets $M_i[s'] := 0^L$, a slot with only zeroes.
2. \mathcal{P}_i commits to $M_i[[1..N]]$, the individual bits of M_i .
3. Through a proof, \mathcal{P}_i must show that M_i has the proper format, i.e. that at least $S - 1$ slots are zero. To this end we use a straightforward subprotocol:
 - i \mathcal{P}_i chooses a random permutation σ of size S , and uses it to permute the slots in M_i , thus creating M'_i . In other words, $M'_i[s] := M_i[\sigma(s)]$. Then he commits to the individual bits of M'_i .
 - ii A random challenge bit c is generated by the trusted source.
 - iii If $c = 0$ then \mathcal{P}_i reveals the permutation σ and proves equality between \vec{M}'_i and \vec{M}_i under the permutation σ . If $c = 1$ then \mathcal{P}_i opens the bit commitments of M'_i for those slots that contain zeroes only.

This protocol must be executed K times in parallel, where K is a security parameter. Cheating succeeds only if \mathcal{P}_i can predict the challenge bits in each round, which happens with probability 2^{-K} .

Commitment and proof of C_i . \mathcal{P}_i now adds the random bits R_i exchanged between his neighbors to the input M_i in order to compute his contribution C_i as follows:

$C_i[[n]] := M_i[[n]] \oplus R_i[[n]] := M_i[[n]] \oplus R_{i_{j_1}}[[n]] \oplus \dots \oplus R_{i_{j_u}}[[n]]$, where j_1, \dots, j_u are the indexes of \mathcal{P}_i 's neighbors, and where n ranges from 1 to N . Then \mathcal{P}_i publishes C_i and signs.

Observe that during the preliminary phase \mathcal{P}_i committed himself to R_{i_j} , and in the first step of the publication phase he committed to M_i , in both cases using the special BCX commitment scheme presented in section 3.2. So using the protocol presented in that section, \mathcal{P}_i can show (in the "committed world") that the assignment of C_i is correct, i.e. that indeed $C_i[[n]] = \overline{M_i[[n]]} \oplus \overline{R_{i_{j_1}}[[n]]} \oplus \dots \oplus \overline{R_{i_{j_u}}[[n]]}$ for each n .

4 Technical Considerations

4.1 Calculating the Collision Probability

Considered separately from the context of voting, the Non-Interactive Dining Cryptographers channel deserves a performance analysis. Since participants choose slots randomly, there always exists a chance that a collision occurs, i.e. two participants occupy the same slot, and consequently the corresponding slot contents (the vs) are lost. If $S = 365$ and $P = 23$, we are back to the birthday paradox: with probability approximately $1/2$ we have a collision, so the message of two participants is lost. To reduce this probability we can increase S . A well-known formula that approximates the collision probability for this case is $1 - e^{-P(P-1)/2S}$.

Another solution is to run Q DC nets in parallel. The probability that in all of them a collision occurs is $(1/2)^Q$, and that the *same* participant is involved in all of them equals $(\frac{2}{P})^Q$ (where we assume that the collisions in the DC nets are independent, and where we ignore collisions which involve more than two participants (which have a very low probability)).

But we can do even better. Instead of using $Q = 10$ (say) parallel nets, it is certainly more effective to use the same total number of slots, i.e. $S' = 3650$ but let the participant choose 10 slots randomly, instead of only one. Since the first version (Q parallel nets) is a special case of the second ($S' = QS$), the collision probability of the second is bounded by the first. Preliminary computer simulations suggest it is orders of magnitudes lower.

Approximating this probability accurately is not a simple exercise and a more careful analysis is appropriate. For instance, it would be interesting to be able to answer questions such as: Given a total of S slots and P participants, how many message T should each participant send in order to maximize successful completion of the protocol? Or reversely, given P participants, how should we choose S and T if we want the failure probability to be really low, say 10^{-20} ? These questions are still subject of ongoing research.

4.2 Optimizing the BCX

The current version of the protocol is rather crude, the main point of this paper being to show that unconditional privacy in voting is possible in a conceptually simple way. Very rough estimates indicate that the current version of this protocol will result in files

in the order of giga- or terabytes, for $P = 500$ (the average size of a precinct in Brazil). However, by fine-tuning of the protocol and a careful analysis of the probabilities, substantial gains can be had.

For instance, a major cause is the expansion caused by the BCX, as every bit of the channel needs at least one BCX, which is very inefficient representation. In fact, we can show that a generalization of the bit commitment scheme used by Bos in his voting protocol ([7], Chapter 3) has exactly the desired properties but with a constant size, resulting in substantial gains. This will be detailed in the final version of this paper. Another possibility for savings is that the current protocol is in some sense too robust, and that by trading off the probability of catching someone cheating on an individual vote some efficiency can be gained.

4.3 How to Prevent Ballot Marking

In the case of voting, a problem is that \mathcal{P}_i can choose the index s , the slot he wishes to occupy, any way he likes. He could therefore abuse this freedom to mark his vote by choosing a particular s , instead of choosing it at random.

The solution is to take away \mathcal{P}_i 's freedom to choose s . Instead, s must be determined in some deterministic, pseudo-random way. As a first approach, one could apply a hash function on v_i , but this property can only be verified after all the contributions have been posted – it would imply that contributions that do not fulfill this property will be declared invalid.

A more general solution is obtained by calculating the hash on a set of known values, such as the BCXs on M_i and on the R_{ij} . For instance, the value $h(\text{BCX}(M_i), \text{BCX}(R_{ij}))$ could fix an arbitrary permutation $\pi \in \{1, \dots, S\}$, which would be used to mix the slots of C_i , i.e. $C_i[\pi(x)] = M_i[x] \oplus R_i[x]$. Again, the protocol in §3.2 would proof equality respecting this permutation π .

5 Conclusions

This paper shows a conceptually simple protocol for voting with unconditional privacy. The paper does so using a non-interactive version of the Dining-Cryptographers protocol, which is not as efficient (in terms of message size) as other voting protocols that offer unconditional privacy, but is of interest in itself since it may have applications other than voting.

The resulting protocol is certainly feasible for voting in small groups where the chance of someone disrupting or not participating is low. Otherwise it might be wiser to define a small number of authorities, whose main role is to reduce the interactions necessary to eliminate no-showers. As in the mix networks, these authorities protect the privacy of the voters, but unlike the mix case, there is no additional computational assumption.

The author strongly believes that unconditional privacy for voting is a desirable property. The fact that at some unknown point in the future voter privacy is completely violated is not acceptable, and the public may actually reject electronic voting systems once this point becomes clear. Therefore, the search for practical voting protocols with

this property is an important challenge. However, it seems that to get unconditional privacy each voter must exchange a sequence of random bits (dispose of a private channel) with other voters or with authorities. For large elections this might be a very difficult to accomplish.

Acknowledgment

I would like to thank Berry Schoenmakers, who gave important feedback on an earlier (and erroneous) version of this paper. Ron Rivest and Madhu Sudan generously shared information on the issues of approximating the collision probability and ballot marking.

References

1. Fujioka, A., Okamoto, T., Ohta, K.: A practical secret voting scheme for large scale elections. In: Zheng, Y., Seberry, J. (eds.) AUSCRYPT 1992. LNCS, vol. 718, pp. 244–251. Springer, Heidelberg (1993)
2. Traoré, J., Arditti, D., Girault, M.: Voting protocols — state of the art and e-poll project, <http://www.francetelecom.com/sirius/rd/fr/-memento/memento20/chap2.html.php>
3. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM* 24(2), 84–88 (1981)
4. Chaum, D.: Blind signatures for untraceable payments. In: CRYPTO 1982, pp. 199–203 (1982)
5. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 103–118. Springer, Heidelberg (1997)
6. Chaum, D.: The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *J. Cryptology* 1(1), 65–75 (1988)
7. Bos, J.: Practical Privacy. PhD thesis, Technische Universiteit Eindhoven, Eindhoven, The Netherlands (1992)
8. Cramer, R., Franklin, M.K., Schoenmakers, B., Yung, M.: Multi-authority secret-ballot elections with linear work. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 72–83. Springer, Heidelberg (1996)
9. Moran, T., Naor, M.: Receipt-free universally-verifiable voting with everlasting privacy. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 373–392. Springer, Heidelberg (2006)
10. Jakobsson, M., Juels, A., Rivest, R.L.: Making mix nets robust for electronic voting by randomized partial checking. In: USENIX Security Symposium 2002, pp. 339–353. USENIX (2002)
11. Crépeau, C., Graaf, J. van de, Tapp, A.: Committed oblivious transfer and private multi-party computation. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 110–123. Springer, Heidelberg (1995)

An Introduction to PunchScan

Stefan Popoveniuc and Ben Hosp

George Washington University - CS Dept.

Washington DC 20052

{poste,bhosp}@gwu.edu

Abstract. PunchScan is a precinct-read optical-scan balloting system that allows voters to take their ballot with them after scanning. This does not violate the secret ballot principle because the ballots cannot be read without secret information held by the distributed authority in charge of the election. In fact, this election authority will publish the ballots for everyone to see, allowing voters whose ballots were incorrectly omitted to complain. PunchScan vote-counting is performed in private by the election authority – who uses their secret information to decode the ballots – but is verified in public by an auditor. In this paper we describe how and why PunchScan works. We have kept most of the description at an outline level so that it may be used as a straw model of a cryptographic voting system.

1 Motivation

The accurate results of a democratic election are at the heart of any modern society. Democracies are built throughout the world with the commitment to have elected individuals representing the entire population of a nation. To be able to record the wish of the people accurately we need to have a voting system that is transparent, reliable and verifiable. We need to be able to prove that the elections are run correctly, that every vote counts, and that the every person going to the polls and exercising their right to vote can make a difference. At the same time, we have to respect the secret nature of any vote. Linking a voter to a vote should not be possible, with or without the complicity of the voter.

PunchScan is a novel voting system and extremely easy to use, both by the voter and by the people running the elections. It is transparent and reliable, and provides public verifiability, election integrity and enhanced voter privacy.

2 Key Elements/Ideas

There are three key elements that make PunchScan work:

1. The ballot is made out of two separate pages. When the two pages are put together, the resulting ballot reveals the choices of the voter. When only one page is viewed, it gives no information – in the computational sense – about what candidates the voter chose. Thus, if one page of the ballot is destroyed, the voter can keep the other page, without violating ballot secrecy.

2. A mechanism allows the recovery of the candidate choices from only one page of the ballot
3. The integrity of the election is provable through pre- and post-election audits.

These ideas are common both to PunchScan and to a previous method of David Chaum's [Cha]. However, PunchScan is more practical, because it does not suffer from the perfect alignment problem of the previous method, because the cryptography used is simpler, and because the time required to find the result and obtain the integrity proof is smaller.

3 High-Level System Design

PunchScan achieves publicly verifiable integrity while maintaining a voter friendly interface using an optical scan-like ballot. It gives each voter the opportunity to take their vote home and check that it is counted in the final tally. In this section, we first describe the ballot itself, then we present all the phases of the voting process as seen by all the participants: voters, the election authority, and candidates.

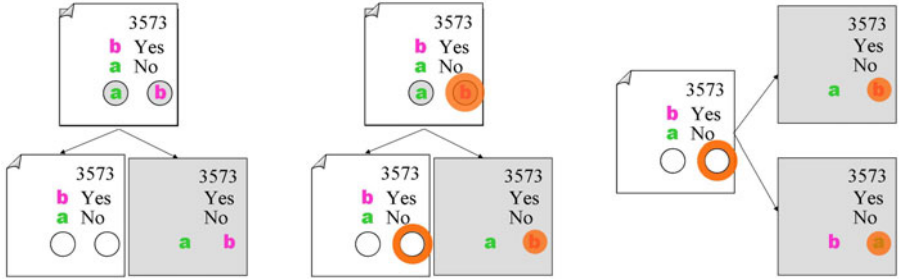
We assume that the candidates are auditing the election, since they are the ones that should care most about a correct outcome; in particular, each candidate would want to check that his rightful votes were not given to another candidate.

3.1 Ballot Design

A ballot consists of two stacked sheets of paper. The top page of the ballot has holes in it, and the information on the bottom page can be read through the holes. Both pages also contain all the text needed on the ballot, such as contests (i.e.: ballot questions) and the candidates' names. On the top page, every answer has a symbol assigned to it and the assignment of symbols to answers varies from ballot to ballot. On the bottom page of the ballot, there is an (apparently) unordered list of symbols and their order differs from ballot to ballot. The top and the bottom ballot pages are aligned in such a way that when they are overlaid, for every question on the ballot, the symbols from the bottom page are visible through the holes made on the top page (see figure I(a)).

In PunchScan, the voter uses a dauber to mark the selection of candidates. A dauber is a pen that leaves a disk of ink on the paper when it makes contact, just like the ones used by Bingo players to mark the numbers on their tickets. The diameter of the ink disc is greater than the diameter of the hole punched through the top page, which means the dauber leaves a mark on both the top and bottom ballot pages. Figure I(b) contains a ballot voted for "Yes".

Because the order of the symbols on the two pages of a ballot is different (and independent), one cannot determine which mark is for which candidate by viewing only one page. We assume that the association of candidates with symbols and the order of the symbols on the bottom page are uniformly random. Figure I(c) has the right answer selected on the top layer; depending on which possible bottom layer is this ballot's actual bottom layer, that mark could represent a vote for "Yes" or a vote for "No", both with a probability of 50%.



(a) A sample ballot. When the two pages are overlaid, the symbols on the bottom page are visible through the holes.

(b) A voted ballot. If you look at each layer individually, you cannot say that the mark is for “Yes” or for “No”.

(c) Given only one layer of the ballot, the marks on that layer are equally likely to represent a vote for any candidate.

Fig. 1. PunchScan’s ballot

3.2 Chronological Description

There are three phases of the voting process:

- the preelection phase (labeled B for *Before*)
- the election phase (labeled E for *Election*)
- the postelection phase (labeled A for *After*)

The preelection phase. The preelection phase is preparatory and allows the setup of the election and integrity proofs. During the preelection phase, the ballots are generated, printed and audited. Also, the information that allows recovering the choice from one page of the ballot is generated and checked. The chronological order is the following:

- B.1 The election authority generates ballots and commits to them.
- B.2 The election authority generates and commits to the information necessary for decrypting one page of the ballot when the other one is destroyed.
- B.3 The candidates challenge the election authority and ask to see some of the ballots (say half), along with the information from B.2.
- B.4 The election authority provides the requested ballots, and opens the commitments associated with them, thus spoiling them.
- B.5 The candidates check to ensure that the commitments are consistent with the opened ballots.

Election day. On election day, the voters go to their assigned polling places, authenticate themselves as legitimate voters, and get a ballot from the election officials.

- E.1 The voter is given a sealed ballot.
- E.2 Without seeing the order of the symbols on either page, the voter commits to the page that will be kept (e.g. by making a special mark on the other page).
- E.3 The voter marks the hole that has the symbol associated with their favorite candidates on the ballot.
- E.4 The voter separates the two pages, destroys the unchosen one and keeps the one chosen in [E.2].
- E.5 The surviving page is scanned, and the positions of the marks are recorded and made public. Henceforth, all references to “ballot” will refer to this surviving page.

In an earlier version, the voter chose which page to keep after seeing and marking their ballot. The early choice of the page to become a receipt is necessary to prevent an attack described by John Kelsey.

The postelection phase. After all the polls close, the election is audited and proofs carried out to ensure the integrity of the election. The chronological order of the events following an election is as follows:

- A.1 Any voter can go to the election authority web site, enter a serial number for her ballot, check that the ballot is there, and that it accurately resembles the page she possesses.
- A.2 The election authority processes all ballots to produce decrypted versions, along with a partially decrypted form of all the ballots.
- A.3 The candidates ask to see some of the transformations from the original ballots to the partially-decrypted forms, and some of the transformation from the partially-decrypted form to the clear form.
- A.4 The election authority replies to the challenges made by the candidates in [A.3].
- A.5 The candidates check to see if the reply of the election authority is consistent with the commitments made in the preelection phase [B.2] and with the information made public in [A.2].

4 Description by Roles

4.1 The Voter

On Election Day, a voter comes to the assigned polling place and authenticates herself as a legitimate voter. She gets a dauber and a ballot, and before seeing it, commits to the page that she will keep. She enters a private voting booth. She chooses her favorite candidates by making a mark with the dauber on the hole that has the symbol associated with her favorite candidate. She then shreds the unchosen page and keeps the other one. Then, she scans the kept page. She may walk out of the polling place with this page, which serves as her (encrypted) receipt. Later, she can go to a web site, type in the serial number of her ballot, and check that the ballot is there. No other checks are required from the voter.

4.2 The Election Authority

In the preelection phase, the election authority decides the format of a canonical ballot. This is the one from which all the other ballot variants will be generated. Also, the canonical ballot is used to recover the choices of the voters, after one page of the ballot has been destroyed.

The election authority generates at least twice the number of ballots needed in the election, and commits to them (making the commitment public; the ballots themselves remain secret). It also generates and commits to information necessary to recover the intent of a voter from one page of the ballot.

In response to the preelection challenge [B.3], the election authority discloses all the information about half (or a significant fraction) of the ballots (thus spoiling them). This allows the candidates to check the commitments and ensures (with high probability) that all the ballots have been correctly generated.

After the election, the election authority posts partially decrypted ballots and cleartext ballots. To prove that both decryptions (partial and final) were done correctly, for each vote the election authority will reveal either how it transformed the voted ballot into a partially decrypted one, or how it transformed a partial decrypted ballot into a cleartext one, but not both for the same ballot. The auditors choose which part will be revealed, and the chances of a cheating election authority being detected grow exponentially with the number of votes cheated on.

4.3 The Candidates

We assume that the candidates are competing in an election. Because of this, we can safely allow the candidates also to play the role of auditors. As auditors, the candidates challenge the election authority during preelection and postelection and check that the replies are consistent with the commitments.

5 An Example

We describe a minimal example: the election consists of a single binary contest; the voters vote “Yes” or “No”. The election authority decides that, in the canonical ballot, the symbol “a” is associated with “Yes” and the symbol “b” with “No” on the top page. The election authority also decides that the order is “a” “b” on the bottom page. The canonical ballot is presented in figure 2(a). The election authority defines what is a shift of one from the canonical form on top and bottom pages. The canonical ballot corresponds to a shift of 0 (call it a non-flipped ballot) and the non-canonical ballot corresponds to a shift of one (call it a flipped ballot). Figure 3(a) contains all the possible top and bottom pages. Any top page can be combined with any bottom page to produce a ballot as seen in Figure 3(b). The four types of ballots are equally likely.

A non-flipped top page combined with a flipped bottom page results in a flipped ballot. All the possibilities are in table 1. Note that we are only interested in knowing if the entire ballot is flipped or not, not individual pages.

To decrypt one page of the ballot, it is necessary to know if it came from a flipped or non-flipped ballot, to know if it should be flipped or not to get the

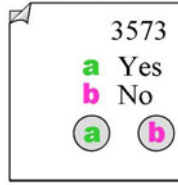


Fig. 2. The canonical ballot for a Yes/No contest

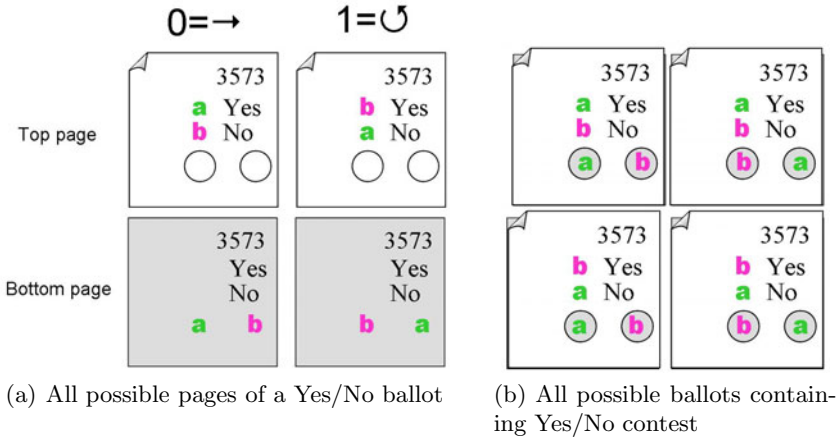


Fig. 3. PunchScan’s ballot

canonical ballot. In PunchScan, this information is split into two flip/non-flip operations (flip1 and flip2) for each ballot. When combined, these operations transform the ballot page to the canonical ballot. The information is split so that one half can be made public for auditing purposes. The relation that has to hold between the pages of the ballot and the information used for recovering is: $top \oplus bottom = flip1 \oplus flip2$.

The election authority makes public commitments to the ballots and to flip1 and flip2. The candidates choose half the ballots at random and the election authority makes public the requested ballots along with the flip1 and flip2 for each ballot. Anyone can check that the equation $top \oplus bottom = flip1 \oplus flip2$ holds. Only the ballots that were not made public in this phase (pre election) will be further used in the election.

Table 1. Flipped / Non Flipped logic

\oplus	Non Flipped	Flipped
Non Flipped	Non Flipped	Flipped
Flipped	Flipped	Non Flipped

During the election phase, the election authority publishes all the marked pages (half ballots) as voted on by voters. After the election, it publishes the intermediary state of the ballots (ballots \oplus flip1) and the decrypted ballots (ballots \oplus flip1 \oplus flip2). These are commitments to the values of flip1 and flip2 used in the decryption of the voted half ballots.

During the postelection phase, the election authority is asked to open either flip1 or flip2 but not both, since opening both would allow the linking of a voted ballot to the corresponding decrypted one. Also, it is necessary that the partially-decrypted ballots and the decrypted ones be in a random order (distinct from each other and from the order of the voted ballots).

The election authority defines the following tables:

- P (for **Print**)
- D (for **Decrypt**)
- R (for **Results**)

The P table is indexed by ballot serial number and contains the top page (P_1), bottom page (P_2), and space for the filled-in vote (to be entered after the election). It also contains commitments to P_1 and P_2 .

The D table contains the first (D_2) and second (D_4) mark permutations (flips), the partially-decrypted vote (D_3) to be filled in during decryption, and information to connect it with the P table (D_1) and the R table (D_5). It also contains a commitment for each row of D , as well as a commitment for columns D_1 and D_2 , and another commitment for columns D_4 and D_5 .

The R table contains the cleartext votes (after postelection decryption).

For example, consider an election with six votes. The clear data in all the tables is in Table 2 (No single person will ever see all of this information.) Before the election, but after the election authority has made the commitments, the tables look as they do in Table 3.

Table 2. *PDR* tables as the election authority sees them, with all the information available. The tables are properly formed, because, for all the ballots, $D_2 \oplus D_4$ correctly represents whether P_2 is a flipped version of P_1 or not. For example, for ballot number 3, on the top page, “a” is associated with “Yes”, and b with “No”. On the bottom page, the order is “ba”, thus P_2 is a flipped version of P_1 . In the D table, in the row corresponding to 3, we have $\rightarrow \oplus \circ = \text{flip}$. For ballot 1, $C_{1,1}$ is a commitment to P_1 , $C_{1,2}$ is a commitment to P_2 and so on.

Ballot ID	P_1	P_2	P_3	CP_1	CP_2
1	ab	ab		$C_{1,1}$	$C_{1,2}$
2	ab	ba		$C_{2,1}$	$C_{2,2}$
3	ba	ab		$C_{3,1}$	$C_{3,2}$
4	ba	ba		$C_{4,1}$	$C_{4,2}$
5	ab	ba		$C_{5,1}$	$C_{5,2}$
6	ba	ab		$C_{6,1}$	$C_{6,2}$

D_1	D_2	D_3	D_4	D_5	DC
6	\rightarrow		\circ	5	C_A
5	\circ		\rightarrow	4	C_B
2	\circ		\rightarrow	1	C_C
1	\circ		\circ	3	C_D
4	\rightarrow		\rightarrow	2	C_E
3	\rightarrow		\circ	6	C_F
$CD_{1,2}$			$CD_{4,5}$		

R_{id}	R_1
1	
2	
3	
4	
5	
6	

Table 3. PD tables in the preelection phase, as the public sees them

Ballot ID	P_1	P_2	P_3	CP_1	CP_2
1				$C_{1,1}$	$C_{1,2}$
2				$C_{2,1}$	$C_{2,2}$
3				$C_{3,1}$	$C_{3,2}$
4				$C_{4,1}$	$C_{4,2}$
5				$C_{5,1}$	$C_{5,2}$
6				$C_{6,1}$	$C_{6,2}$

D_1	D_2	D_3	D_4	D_5	DC
					C_A
					C_B
					C_C
					C_D
					C_E
					C_F
$CD_{1,2}$			$CD_{4,5}$		

Table 4. PD tables after the election authority has replied to the request to open ballots 2, 4, and 5

Ballot ID	P_1	P_2	P_3	CP_1	CP_2
1				$C_{1,1}$	$C_{1,2}$
2	ab	ba		$C_{2,1}$	$C_{2,2}$
3				$C_{3,1}$	$C_{3,2}$
4	ba	ba		$C_{4,1}$	$C_{4,2}$
5	ab	ba		$C_{5,1}$	$C_{5,2}$
6				$C_{6,1}$	$C_{6,2}$

D_1	D_2	D_3	D_4	D_5	DC
					C_A
5	⊙		→	4	C_B
2	⊙		→	1	C_C
					C_D
4	→		→	2	C_E
					C_F
$CD_{1,2}$			$CD_{4,5}$		

The candidates challenge the election authority to open a random half of the ballots, say the ones numbered 2, 4 and 5. The election authority reveals the requested information, and the tables look as they do in Table 4. Ballots 2, 4, and 5 now cannot be used in the election and are excluded from any further representation of the tables (see Table 5).

Assume that the voters mark their ballots as follows: on ballot 1, the left mark is marked, and the top page is chosen; on ballot 3, the right mark and the bottom page are chosen; on ballot 6, the left mark and the top page are chosen. Because the canonical ballot is “ab”, “ab” (that is, “ab” on both pages), left is associated with “a”, and right with “b”. The voters’ choices eventually end up in P_3 , and when they do, each row describes what can be learned through knowledge of the ballot page chosen by the voter.

The election authority performs the first flip to ballots 1,3 and 6 to obtain the partially decrypted ballots as in D_3 , and the totally decrypted ballots as in R_1 (see Table 6). The ballots in both D and R are shuffled independently, so it is not possible to link rows among tables P , R and D . During the postelection phase, the auditor asks the election authority to open either the left or the right side of D (but not both). If the election authority cheats, the auditor will catch it with probability 0.5 (for a higher probability see section 6.4). In our example, suppose the auditor chooses the right side. The election authority then reveals D_4 and D_5 . The auditor can now check that $D_3 \oplus D_4 = R_1$, and that the commitment $CD_{4,5}$ to the columns D_4 and D_5 is valid.

Table 5. Ballots that can be used by voters on election day. The other ballots were spoiled during the preelection phase. The row commitments are not shown anymore because they won't be checked, since no other complete row will ever be opened.

Ballot ID	P_1	P_2	P_3
1			
3			
6			

D_1	D_2	D_3	D_4	D_5
$CD_{1,2}$			$CD_{4,5}$	

Table 6. *PDR* snapshot after the polls close. One cannot say what row in the D table corresponds to what row in the P or R table, because the rows are shuffled. Thus, the secret ballot principle is satisfied.

Ballot ID	P_1	P_2	P_3
1	ab		a
3		ab	b
6	ba		a

D_1	D_2	D_3	D_4	D_5
		a		
		b		
		b		
$CD_{1,2}$			$CD_{4,5}$	

R_{id}	R_1
3	a
5	b
6	a

Table 7. *PDR* snapshot after the postelection audit. The election authority was asked to open the right side of the D table. Anyone can check that the partially decrypted result transformed by D_4 gives the result in R ($D_3 \oplus D_4 = R$), so the election authority did not cheat. Also $CD_{4,5}$, the commitment to D_4 and D_5 , is checked. Note that there is still no link between P and R , so privacy is maintained.

Ballot ID	P_1	P_2	P_3
1	ab		a
3		ab	b
6	ba		a

D_1	D_2	D_3	D_4	D_5
		a	⊙	5
		b	⊙	3
		b	⊙	6
$CD_{1,2}$			$CD_{4,5}$	

R_{id}	R_1
3	a
5	b
6	a

6 A More Technical Description

This section provides a more technical description of PunchScan.

6.1 The Ballot

Let S be a set of symbols. The symbols in S will appear on both the top and bottom page. We assume that S is sorted and the order is fixed. We denote by “canonical ballot” a ballot that will have S printed in order on both the top and bottom page. Let T_p (top permutation), B_p (bottom permutation), and D_2 be three random, independent permutations of S (in an implementation, the permutation would be pseudorandomly generated as described in section [A](#)).

Compute D_4 such that $B_p \circ T_p^{-1} = D_2 \circ D_4$. Therefore, $D_4 = D_2^{-1} \circ B_p \circ T_p^{-1}$.

6.2 The Tables

We describe the *PDR* tables using notation from relational algebra, a system of notation heavily used in databases. It has the notions of relations (tables), projections (π - SQL SELECT), selection (σ - SQL WHERE) and join (\bowtie). A relation $R(A, B), A \rightarrow B$ means that A implies B (given A , B is uniquely identified). A is called a key of relation R .

Let P (print) be the following relation:

$$P(B_{id}, P_1, P_2, P_3, CP_1, CP_2), B_{id} \rightarrow (P_1, P_2, P_3, CP_1, CP_2)$$

where B_{id} is the ballot id (the serial number of the ballot), P_1 is T_p , P_2 is B_p , P_3 is a projection of $B_p o T_p^{-1}$ (voter choices), CP_1 is a commitment to P_1 , and CP_2 is a commitment to P_2 . The commitments are cryptographic commitments (see Section [B.2](#) for details). P contains $2n$ records.

Let D (decrypt) be the following relation:

$$D(D_1, D_2, D_3, D_4, D_5, DC), D_1 \rightarrow (D_2, D_3, D_4, D_5, DC)$$

where D_1 is a foreign key pointing to the B_{id} attribute of P , D_5 is a foreign key pointing to the R_{id} attribute of R (see below), D_2 and D_4 are permutations of S described above, D_3 is $P_3 o D_2$, and DC is a commitment to the tuple (D_1, D_2, D_4, D_5) . D contains $2n$ records.

Let CD (commitments to the columns of D) be the following relation:

$$CD(CD_{1,2}, CD_{3,4})$$

This relation has only one record. $CD_{1,2}$ is a commitment to D_1 and D_2 ; $CD_{4,5}$ is a commitments to D_4 and D_5 .

Let R (results) be the following relation:

$$R(R_{id}, R_1), R_{id} \rightarrow (R_1)$$

where R_{id} is a unique identifier and R_1 is $P_3 o D_2 o D_4$. R contains $2n$ records.

To select all the information for a ballot, we write:

$$(P \bowtie_{B_{id}=D_1} D) \bowtie_{D_5=R_{id}} R$$

6.3 The Timeline

Before the election the election authority computes $P(B_{id}, P_1, P_2, CP_1, CP_2)$, $D(D_1, D_2, D_4, D_5, DC)$, $CD(CD_{1,2}, CD_{4,5})$ and makes public $P(B_{id}, CP_1, CP_2)$, $D(DC)$ and $CD(CD_{1,2}, CD_{4,5})$.

In the preelection audit, the auditor randomly selects half of the records in P . The election authority reveals $P \bowtie_{B_{id}=D_1} D$ for all the requested records. The auditor can check that $B_p o T_p^{-1} = D_2 o D_4$. and that the commitments CP_1 , CP_2 , and DC are valid.

During the election, the voters fill in P_3 .

After the election, the election authority computes $D_3 = P_3 o D_2$ and $R_1 = D_3 o D_4$ and makes D_3 and R_1 public.

In the postelection audit, the auditor asks the election authority to either reveal (D_1, D_2) or (D_4, D_5) , but not both. The election authority reveals the requested information. The auditor can either check that $P_3 o D_2 = D_3$ (using $P \bowtie_{B_{id}=D_1} D$) or $D_3 o D_4 = R_1$ (using $D \bowtie_{D_5=R_{id}} R$). The chance of the election authority cheating and not being caught is 50% (see section 6.4). $CD_{1,2}$ and $CD_{4,5}$ are also checked.

6.4 Multiple Instances of D

Because the election authority can cheat with 50% probability of success (i.e., nondetection), we introduce multiple instances of D . In other words, we modify the relation D as follows: Let D (decrypt) be the following relation:

$$D(i, D_1, D_2, D_3, D_4, D_5, DC), (i, D_1) \rightarrow (D_2, D_3, D_4, D_5, DC)$$

where i is the instance number and the rest is as described in Section 6.2

Let CD (commitments to the columns of D) be the following relation:

$$CD(i, CD_{1,2}, CD_{3,4}), i \rightarrow (CD_{1,2}, CD_{3,4})$$

where i is a foreign key pointing to the i attribute of D .

In the postelection audit, we can now make k challenges, where k is the number of D instances. The auditor will ask to open either (D_1, D_2) or (D_4, D_5) for each instance of D . The chance that the election authority cheats successfully is one out of 2^k . We can make this probability arbitrarily small by increasing k .

6.5 Multiple-Question Ballots

We have been implicitly assuming that there is only one question per ballot. The situation becomes slightly more complicated if this is not the case. PunchScan works just fine for multiple-question ballots but the decrypted ballots will preserve the “cross-question” relationships: for example, if 90% of the people who voted for Alice for Governor also voted for Bob for President, the results will reflect this. However, PunchScan can be extended to hide these correlations if desired.

Trivially, of course, if PunchScan works for one-question elections then we can conduct an n -question election by giving each voter n one-question ballots. If we want to preserve the cross-question relationship among two or more questions (perhaps if someone voted “No” for a recall election then they are not allowed to vote for a replacement candidate) then we could group those questions on the same ballot. This would work but seems to us to be not as good (from a ballot design, system overhead, and printing cost point of view) as the case when we are using one ballot and running one election.

However, we can readily modify this scheme to fix this problem. Suppose we are running n one-question elections. That is, each voter receives one ballot for each of n elections and votes, then the votes are counted separately for each election. In this situation, there is one P -table and one set of D -tables (and associated R -table) for each of the n elections. Let us note that there is no

information contained in the D -tables for election A that can be used to decrypt the ballots for election B . Since the shuffles for each election are also independent, we do not need to obscure the link between voter x 's encrypted ballots in election A and B , because when they are decrypted the shuffling will obscure the cross-question relationship for us. In other words, we can print these ballots together, on the same piece of paper, with the same serial number (and the same P -table row), just as in the original scheme that reveals the correlations. Because the ballots are decrypted separately, this does not provide any more information regarding the cross-question relationships.

7 Proofs

This section contains proofs of some security properties of PunchScan.

7.1 Privacy

In this context, the maintenance of privacy requires that an observer's probability distribution of the contents of a given ballot i (i.e.: the value of voter i 's vote) be unchanged by observation of the cryptographically-hidden data. In other words,

$$p(b_i|PDR) = p(b_i|R),$$

where b_i is the value of ballot i , PDR is the entire publicly-observable ballot data matrix, and R is the results column of that matrix.

Attacks on P . The most straightforward way for an attacker to use the secret parts of PDR to reveal the vote of voter i would be to simply decrypt $P_{1,i}$ and $P_{2,i}$ and use those to decode $P_{3,i}$. If the attacker is unable to break this cryptography, then learning P would not affect his probability distribution on b_i . This cryptography can be made arbitrarily strong in order to protect privacy at any desired level of computational security.

Attacks on D . Another method would involve an attack on the shuffle; that is, decrypting the unrevealed link between P and D (D_1) or between D and R (D_5). However, the same cryptography is used to secure those columns of D , so again, an attacker unable to break the cryptography could not learn anything useful from D .

7.2 Integrity

There are four elements of the PunchScan process that are vulnerable to some extent to manipulation of the vote tally by the election authority.

- The ballots may be improperly formed.
- The ballots may be improperly printed.
- The ballot markings may be improperly recorded.
- The marked ballots may be improperly decrypted.

Each of these vulnerabilities is addressed by an audit procedure.

7.3 First Audit

The first audit procedure ensures that the ballots are well-formed, meaning that for each ballot, $P_1 \oplus P_2 = D_2 \oplus D_4$ for the row in each D -matrix associated with that ballot. This involves spoiling some fraction of the ballots by unlocking this secret data.

In general, suppose there are n ballots, the election authority has cheated by malforming k of them, and f ballots are chosen at random to be examined. The probability that the election authority gets away with this attack is the number of possibilities where the auditor chooses only valid vote divided by the number of all possible choices.

The number of all the possible choices is $\binom{f}{n}$ (n choose f). The number of ways to choose f valid ballots from a total of n ballots where k of them are invalid, is $\binom{f}{n-k}$ (choose f votes out of $n-k$ that are valid). So the election authority cheats successfully with the following probability:

$$p = \frac{\binom{f}{n-k}}{\binom{f}{n}} = \frac{\frac{(n-k)!}{f!(n-k-f)!}}{\frac{n!}{f!(n-f)!}} = \frac{\frac{(n-k)!}{(n-k-f)!}}{\frac{n!}{(n-f)!}}$$

Note that $f + k < n$, so that $n - k - f > 0$ and $(n - k - f)!$ exists and is not the special case $0!$. If $f + k > n$ then the probability is 0.

In the interest of simplicity, from here we may compute two upper bounds on the chance that this attack will not be detected:

$$\begin{aligned} \frac{(n-k)!}{n!} \times \frac{(n-f)!}{(n-k-f)!} &= \frac{(n-f) \times (n-f-1) \times \dots \times (n-f-k+1)}{n \times (n-1) \times \dots \times (n-k+1)} \\ &= \frac{n-f}{n} \times \frac{n-f-1}{n-1} \times \dots \times \frac{n-f-k+1}{n-k+1} \\ &= \left(1 - \frac{f}{n}\right) \times \left(1 - \frac{f}{n-1}\right) \times \dots \times \left(1 - \frac{f}{n-k+1}\right) \\ &< \left(1 - \frac{f}{n}\right)^k \\ \frac{(n-k)!}{(n-k-f)!} \times \frac{(n-f)!}{n!} &= \frac{(n-k) \times (n-k-1) \times \dots \times (n-k-f+1)}{n \times (n-1) \times \dots \times (n-f+1)} \\ &= \frac{n-k}{n} \times \frac{n-k-1}{n-1} \times \dots \times \frac{n-k-f+1}{n-f+1} \\ &= \left(1 - \frac{k}{n}\right) \times \left(1 - \frac{k}{n-1}\right) \times \dots \times \left(1 - \frac{k}{n-f+1}\right) \\ &< \left(1 - \frac{k}{n}\right)^f \end{aligned}$$

Thus, our upper bound on the probability that the election authority gets away with malforming k out of n ballots when f of those ballots are audited is $\min[(1 - \frac{f}{n})^k, (1 - \frac{k}{n})^f]$.

7.4 Second Audit

In order to check that a given ballot receipt was properly printed, one can re-encrypt it (that is, recompute the commitments) and compare it with the P -matrix. Suppose n ballots remain unspoiled after the first audit, f are actually used by voters who later check the commitments, and k of them are improperly printed. Once again, the upper bound on the probability that none of the misprinted ballots are detected is $\min[(1 - \frac{f}{n})^k, (1 - \frac{k}{n})^f]$.

7.5 Third Audit

In addition to checking that the ballot is correctly printed, one can also verify that the recorded ballot mark matches the mark on the receipt. In effect, this verifies the correctness of P_3 . Again, if n ballots remain unspoiled after the first audit, f are actually used by voters who verify that their ballot marks are correctly recorded online, and k ballots are incorrectly recorded, then the upper bound on the probability that none of the incorrectly-recorded marks are detected is $\min[(1 - \frac{f}{n})^k, (1 - \frac{k}{n})^f]$.

7.6 Fourth Audit

The election authority may influence the vote tally by incorrectly decrypting the ballots. There are two methods we may use for auditing the election authority to ensure that this does not occur.

Ballot-wise Auditing. Suppose the auditor goes through a D -matrix ballot-by-ballot (that is, row-by-row) and randomly chooses whether to inspect (open) the “left” or “right” commitment for each ballot. This situation is different from the first three audits because all ballots are inspected, but each inspection has only a $\frac{1}{2}$ chance of catching a modification. This makes the situation simpler; the chance of k modified ballots all escaping detection is 2^{-k} .

Table-wise Auditing. On the other hand, the auditor may choose to open all the “left” or “right” commitments for a given D -matrix. Assuming that the election authority intends to cheat during the decryption and is aware of this, he will put all his cheating in a given D -matrix in either the “left” or “right” commitment, so that he has a $\frac{1}{2}$ chance of escaping detection when that D -matrix is inspected. If there are n D -matrices, then the chance of escaping detection if any ballots are incorrectly decrypted is 2^{-n} .

Comparison. Both of these methods have desirable properties. The ballot-wise method has the feature that the probability of detecting cheating is a function of the number of ballots cheated on, and increases exponentially with a linear increase in number of cheated ballots. The table-wise method has the feature that the audit does not reduce the size of the anonymity set created by the shuffle.

8 Related Work

Verifiable electronic voting has been introduced by David Chaum in 1981 [Cha81]. The first voter verifiable version used visual cryptography [CvdGRV07]. Peter Ryan introduced the candidate permutation idea and developed an improved ballot, much more usable and implementable in a voting system called Pret-A-Voter [CRS05]. An early stage of PunchScan was analyzed by John Kelsey [JK07] who came up with an attack based on the fact that the voter can see the ballot and then decide which page to keep (see Section 3.2).

Acknowledgments

We would like to thank David Chaum, Poorvi Vora, Rick Carback, Jeremy Robin and Ben Adida, for the vibrant discussions and insightful comments.

References

- [Cha] Chaum, D.: Secret ballot receipts and transparent integrity - better and less-costly electronic voting at polling places, <http://www.vreceipt.com/article.pdf>
- [Cha81] Chaum, D.L.: Untraceable electronic mail, return address, and digital pseudonym. Communication of ACM (February 1981)
- [CRS05] Chaum, D., Ryan, P.Y.A., Schneider, S.: A practical voter-verifiable election scheme. In: di Vimercati, S.d.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 118–139. Springer, Heidelberg (2005)
- [CvdGRV07] Chaum, D., van de Graaf, J., Ryan, P.Y.A., Vora, P.L.: Secret ballot elections with unconditional integrity. Technical report, IACR Eprint (2007), <http://eprint.iacr.org/> or <http://www.seas.gwu.edu/~poorvi/cgrv2007.pdf>
- [JK07] Moran, T., Chaum, D., Kelsey, J., Regenscheid, A.: Hacking paper some random attacks on paper based e2e systems (2007), <http://kathrin.dagstuhl.de/files/Materials/07/07311/07311.KelseyJohn.Slides.pdf>

Appendix

A Permutations

PunchScan requires two types of permutations to be generated:

- row permutations
- mark permutations

Row permutations refer to the permutations of the rows of the D table and mark permutation refer to the order in which the positions are associated with marks on the ballot and to D_2 and D_4 .

A.1 Row Permutations Generation

Consider an “unshuffled” D -matrix where $D_1 = [1, 2, \dots, 2n]$, so row x of PDR represents ballot x across the entire row, and D_5 is blank. The election authority should generate this matrix as the first step; call it δ . Generating the row permutations will therefore take the form of the generation of $D^1 \dots D^{n_D}$, where D^i denotes the i^{th} shuffled D -matrix.

The D -matrices will be generated from δ as follows:

1. Randomly shuffle the rows of δ ; call this D^1 .
2. Let D_5^1 equal a random shuffling of $\{1, 2, \dots, 2n\}$.
3. For each i from 2, 3, \dots , $2n$, let D^i equal a random shuffling of the rows of D^1 .

This involves $n_D + 1$ permutations of $\{1, 2, \dots, 2n\}$. It should be clear that if $(y, D_1^i) = x$ and $(y, D_5^i) = z$, then for all j , $(y, D_1^j = x)$ implies that $(y, D_5^j) = z$; in other words, since each row of D^1 contains a pointer to a (unique) row (ballot) of P in D_1 and a (unique) pointer to R in D_5 , reordering its rows does not change the destination (in R) of any ballot in P .

A.2 Implementation

Permutation Algorithm. We use the following permutation algorithm to permute the unshuffled matrix. This algorithm generates a permutation π of $1, 2, \dots, m$, given as input m , some encryption function E , and some key K .

First, create a table with m rows and 2 columns. Populate column 1 of the table with $1, 2, \dots, m$ and column 2 of the table with $E_K(1), E_K(2), \dots, E_K(m)$; in other words, $(i, 2) = E_K((i, 1))$ for every row i . Next, sort the table according to column 2. Let $\pi(i) = (i, 1)$; column 1 is now a permutation of $1, 2, \dots, m$.

If the key K were generated randomly, and the function E is a good encryption algorithm, then the permutation output by the algorithm will be random. (That is, it will preserve any randomness in K .)

Application of the Algorithm. The election authority can use this algorithm to implement the D -matrix generation algorithm above as follows:

1. Generate a permutation π_{D^1} of $1 \dots 2n$. Let $D_1^1 = \delta$, sorted by π_{D^1} ; that is, row x of δ becomes row $\pi_{D^1}(x)$ of D^1 .
2. Generate a permutation π_R of $1 \dots 2n$. Let $D_5^1 = \pi_R$.
3. For each i from 2 to n_D , create D^i by generating a permutation π_{D^i} of $1 \dots 2n$. Let row y of D^1 become row $\pi_{D^i}(y)$ of D^i .

A.3 Mark Permutations

The mark permutations, in contrast, are much simpler to generate. In order to produce all possible associations of candidate names with ballot symbols, it is not necessary to randomly permute both lists; it is only necessary to cyclically shift both lists a (different) random amount. So to generate the mark permutations for ballot x , where the ballot has c candidate names on the top page and c mark

symbols on the bottom page, the election authority only needs to generate two random numbers between 1 and c , and record these numbers as P_1 and P_2 to indicate the shift distance for the pages of ballot x .

Each D -matrix instance will require its own set of decrypting mark permutations (columns D_2 and D_4). (It is for this reason that at least the decrypting mark permutations must be performed after the row permutations.) For each row of each D^i , the election authority generates a random number between 1 and c , and records this number in D_2^i . D_4^i is set such that the modular sum of the ballot's entries in P_1 and P_2 equals the sum of its entries in D_2 and D_4 .

Random Number Generation. The permutation algorithm described above can also be used for the random number generation. The election authority can compute a permutation π of $1, 2, \dots, c$ and use $\pi(1)$ as the random number.

B Commitments

This section describes how the commitments in PunchScan are computed. We use the comma (“,”) to represent the concatenation operation. There are two secret AES 128-bit keys, MK_1 and MK_2 , and a public 128-bit constant, C .

B.1 Computing AES Keys

This section requires the use of two 128-bit AES keys. Given message M , let M_{128} be the first 128 bits of M (if M is shorter than 128 bits, M will be padded with trailing zeros); a random key SK_m is generated as follows:

$$SK_m = D_{MK_1}(C \oplus E_{MK_2}(C \oplus E_{MK_1}(M_{128})))$$

where \oplus is the XOR operation and E and D are AES Encrypt and Decrypt EBC NoPadding operations.

B.2 Commitment Algorithm

Given a message M , the commitment to M is computed as follows:

1. Generate a 128-bit AES key K_m as described in Section [B.1](#).
2. Encrypt the public constant C with K_m , using AES 128-bit ECB NoPadding. Let the result be $SK_m = AES_{K_m}(C)$. Note that SK_m has 128 bits.
3. Concatenate M with SK_m and hash everything using SHA256, resulting in h_1 . So, $h_1 = SHA256(M, SK_m)$.
4. Let $h_2 = SHA256(M, AES_{SK_m}(h_1))$, where the AES encryption is AES 128bit ECB PKCS#5Padding.
5. The commitment is h_1, h_2 (h_1 concatenated with h_2).

We now describe the computation of M for all the commitments needed in PunchScan.

M for P_1 . M is obtained by concatenating the serial number of the ballot to a constant particular to P_1 and with the text on the top page of the ballot. $M = i, "P1", P_1$ where i is a string representing the serial number of the ballot, "P1" is a constant string (capital "P" concatenated with digit "1") and P_1 is the string in P_1 (the string representation of the top page).

M for P_2 . M is obtained by concatenating the serial number of the ballot to a constant particular for P_2 and with the text on the bottom page of the ballot. $M = i, "P2", P_2$ where i is a string representing the serial number of the ballot, "P2" is a constant string (capital "P" concatenated with digit "2") and P_2 is the string in P_2 (the string representation of the bottom page).

M for rows in D . M is obtained by concatenating all the known values in a row in D . The known values are: the pointer to the P table (D_1), the first mark permutation (D_2), the second mark permutation (D_4) and the link to the R table (D_5).

$M = D_1, D_2, D_4, D_5$, each D_i being the string representation of a field in D .

M for columns in D . M is obtained by concatenating all the values in the first column and then concatenating all the values in the second column.

For the leftmost columns:

$M = D_{1,1}, D_{2,1}, D_{3,1}, \dots, D_{n,1}, D_{1,2}, D_{2,2}, D_{2,3} \dots D_{n,2}$

For the right most columns:

$M = D_{1,4}, D_{2,4}, D_{3,4}, \dots, D_{n,4}, D_{1,5}, D_{2,5}, D_{2,5} \dots D_{n,5}$

We only need to protect two 128-bit AES keys, MK_1 and MK_2 , in order to preserve the privacy of the system. The keys can be distributed and recreated as needed, only when a certain threshold of the participants is present.

Note that the public cannot verify that the AES keys have been generated in this way, or rather in some other way. Therefore, this system unfortunately introduces a potential covert channel via the AES keys.

Component Based Electronic Voting Systems

David Lundin

University of Surrey, Guildford, Surrey, GU2 7XH, UK
d.lundin@surrey.ac.uk

Abstract. An electronic voting system may be said to be composed of a number of components, each of which has a number of properties. One of the most attractive effects of this way of thinking is that each component may have an attached in-depth threat analysis and verification strategy. Furthermore, the need to include the full system when making changes to a component is minimised and a model at this level can be turned into a lower-level implementation model where changes can cascade to as few parts of the implementation as possible.

1 Introduction

It appears that each researcher in the field of Electronic Voting Systems contributes to some particular aspect but re-builds the whole system when they wish to implement this rather specific contribution. The idea presented in this paper is that in order to build an e-voting system we simply add certain distinct pieces together — and in order to improve on a particular system we swap one distinct piece for another that fits into the same slot. In short, we are proposing that we start thinking about electronic voting systems as being component based.

A benefit of this thinking is that for each component slot, i.e. a place in a layer where a component of the system can be slotted in, it is possible to define the method of assessing the computational complexity of that component as well as performing a threat analysis. Similarly, if we agree that all components of an electronic voting system should be verifiable and/or auditable then it is possible in this configuration to define for each component a method of verification or audit. When an author then makes changes to one or more components it is possible to in effect “re-run” checks on those components or to employ the same verification method on a different component.

1.1 Domains Captured

As the reader goes through the description of the different layers, she may realise that the components in those layers are not strictly components from a computer systems design perspective. Instead, the components capture the full spectra of the design and implementation of an electronic voting system — in other words they capture the following domains:

- Computer system domain
- Human (user/voter) domain
- Legislative domain

We do not make distinctions between components in the hierarchy that we propose, that is to say that we do not treat components from different domains differently. To a developer of electronic voting schemes this will be perfectly natural as her overview of the system is total. When she considers all the components then they must all fall into place and completely make up the (correct) system. In fact, it is the duty of the developer to ensure that a component that is put forward does fulfill all the requirements on that component.

Others may look differently on the component based electronic voting system. A programmer (implementer) may look only on those components that are implementable in software. Similarly someone who might be consulted on the legal implications on the configuration of an electronic voting system may only consider components from the legislative domain. Both these non-developer persons are examples of people who must be able to trust that the system as a whole depends on each of their respective components and that the configuration of their components are reflections of the requirements and conform to the restrictions of those same components.

It is encouraged that a developer of electronic voting systems that conform to the proposed component based methodology considers the subset of components that are to be implemented in code and makes available to programmers some model that binds these together in some lower-level modelling scheme — this is out of the scope of this paper but seems fairly straightforward in any (favourite) modelling language.

1.2 Cascading Changes

When a system is considered to be component based we believe that changes can easily cascade down the different layers, all the way down to the implementation. In a trivial sense this might simply be that when the developer changes the structure of a component this automatically becomes a list of changes for the implementer to make in the actual code. This is likely to be the scope of other papers in the modelling domain. However, it is easy to see that changes made to one component will only result in changes to the implementation of that same component and not to surrounding or distant components, restricting the work needed to implement the changes.

2 Component Hierarchy

We suggest that an electronic voting system is made up of parts from four comparatively separable and distinguishable layers, each of which builds on the services provided by a lower level layer. We propose to name these layers in the following fashion:

1. Human layer
2. Election layer
3. Computational layer
4. Physical layer

The physical layer enables the reading in of voter choices and the transfer and publishing of the same. The computational layer contains that which is chiefly encapsulated in software and which relies on the physical services of the lower level. The election level encapsulates all those options and configurations relating to the election system being implemented by the two lower levels. Finally the human layer deals with those aspects of the electronic voting system which face the voters.

Table 1. Layers and components

Layer	Components
Human layer	Voter registration Ballot form configuration Polling station layout and management Verifiability front-end
Election layer	Election method Election management system Voter-ballot box communication channel
Computational layer	Cryptography scheme Anonymisation strategy Tallying procedure
Physical layer	Hardware authentication method Publishing strategy Transfer method

The layers and components are shown in Table 1. We will now go through these layers from the bottom up so as to first provide a solid foundation and then explain how all the aspects of electronic voting systems may fit into this model.

2.1 Physical Layer

The physical layer is of course the most basic layer as it supports all other layers with a physical infrastructure to facilitate different hardware based aspects of the system. We divide the physical layer into the following components:

Hardware authentication method. The different physical components of the electronic voting system must be identified using some authentication strategy, for example an asymmetric key pair and a public key infrastructure (PKI) with established trust among voters.

Publishing strategy. Voter verifiable electronic voting systems depend on the information that voters need to perform the verification being delivered to them in a way that they can trust. An observation is that a single source may be an

unreliable publishing strategy but a combination of web sites and newspapers and other independent outlets may be more reliable.

Transfer method. All electronic voting systems capture some information from the voters and transfer it to some repository, be it central or distributed, before the information is interpreted and tallied. In this slot fits methods for transporting such digital information from polling stations to such repositories, for example Secure Socket Layer (SSL) over the Internet or storage on flash drives that are manually distributed.

2.2 Computational Layer

In the computational layer are defined all the components that are performed implicitly by software. These include the following components:

Cryptography scheme. The cryptography scheme employed probably underpins much of the operation of other components in this layer so there is an intra-layer dependence. However, the cryptography scheme is more easily defined in its own component after which the other components of this layer (and others that may rely on it, although restricting the use of the cryptography scheme to the computational layer only would greatly simplify the definition of a scheme in this model) may refer to it. One example of this may be where the cryptography scheme employed is Elgamal and re-encryption mix networks are used as anonymisation strategy in that component.

Anonymisation strategy. We believe that electronic voting schemes that are both “receipt free” and voter verifiable must use some anonymisation strategy to break the link between an encrypted receipt and the plain text vote. This strategy is captured in this component although it most likely is heavily dependent on the cryptography scheme defined in the previous. However, in some cases the anonymisation scheme may rely on “any” cryptographic scheme and so changing the cryptography scheme component does not necessarily interfere with the anonymisation strategy component. One example of where this is true is where the anonymisation strategy is a re-encryption mix network. It is logical that the cryptography scheme may be Elgamal or Paillier and one scheme may be removed and the other slotted in without this requiring the anonymisation strategy to change.

Tallying procedure. The tallying procedure component is simply an encapsulation of the procedure and software that performs the tallying of the decrypted votes. This may seem like a trivial component at first glance but when the component properties that we introduce below are considered it appears that a verification strategy and computational complexity analysis for example is done better across this component than across some external “election administration” software.

2.3 Election Layer

This layer is very much derived from the laws that govern the election. It seems likely that when this layer is implemented it may result in some components not being turned into code and others may be external software. As an example of the latter we can take that the implementation of a component may actually be a formal specification of a piece of software that can successfully audit an election. This is then published and any number of interested parties may write their own piece of software.

The election layer consists of the following components:

Election method. This component contains a specification of the election method that is used and into it is placed simply as much information about the election that is available. If the model is scrutinised as a whole it may be beneficial to have in one part of it the specification of the election at such a high level.

Election management system. All schemes require an election to be set up by some clerks or civil servants — or politicians or some trusted third party. This component is, much like the previous, incorporated into the model for completeness although it does require to communicate with other components.

Voter-ballot box communication channel. This may seem like a much more low-level component but in fact it is included in this layer to enable the high-level definition of the secrecy of the election. In other words it is possible to describe here the full procedure that the voter goes through to cast a vote in a way that is understood not only by the developers of the system but also the general public. This definition may then be used as reference when other components at lower levels in the model are composed.

2.4 Human Layer

Although some components in the previous layer have become human readable and in fact only serve to further the understanding of the lower layers (or, arguably, define the lower layers in a human readable way which then cascades down) this layer deals with all those aspects of the electronic voting system that are facing the voter. Therefore the components we expect to find in this layer are:

Voter registration. The procedure for and timing of voter registration along with the criteria that makes a voter eligible to register are all enshrined in the law relevant to the use of the implementation of the modelled system. They are included in the model in this component so as to provide completeness. As mentioned in Section [1](#) the completeness of the system is entrusted to the electronic voting system developer and so a change in this component must trigger any necessary changes to other components — and these may of course be components that are eventually implemented in code.

Ballot form configuration. The ballot form configuration is very important in any electronic voting system and it seems likely that this component has links to other components, for example the cryptography component in the computational layer.

Polling station layout and management. This is another component that is directly derived from the applicable national and regional legislation. If the component is precise, by which we mean it is directly derived from legislation which sets out exactly the layout and management, then this may dictate the configuration of other components in the model. If it on the other hand is permissive then this would imply that only a few guidelines are given and that if necessary the layout and management of polling stations may be adapted to suit the electronic voting system.

Verifiability front-end. To present a unified verifiability front-end to voters and concerned groups, this components acts to capture the requirements of such a front-end at a high level and thus restrict other components that must comply therewith.

2.5 Component Interaction

As has now become abundantly clear it is impossible to make all components of an electronic voting system fully autonomous — and we simply are not striving to do that. Instead, by making each component as distinct and autonomous as possible and then providing links between components we can reach some compromise between all the good properties brought by a component based model and the necessity of component interaction.

The links we propose are not communication links as such, in fact we regard the communication between different implemented components to be modelled at a lower level. This is supported by the fact that only some components at this level of abstraction are implemented and only some of those communicate — in fact some of these components become entangled in an implementation, for example the cryptography component which is used in an implementation of the anonymisation strategy component in the computational layer.

The component links that we propose are of only two kinds and each link is a mono-directional vector. The link carries some fact attached to it and it either defines a restriction by one component on another or a permission given by one component to another.

The links are fairly simplistic and can easily be illustrated in a graph, and example of which is shown in Figure [11](#). If the underlying fact or facts are coded in some way (perhaps just a unique numerical reference) within the imposing component then the link can easily be shown in a graph simply using an arrow with a name corresponding to that code (reference).

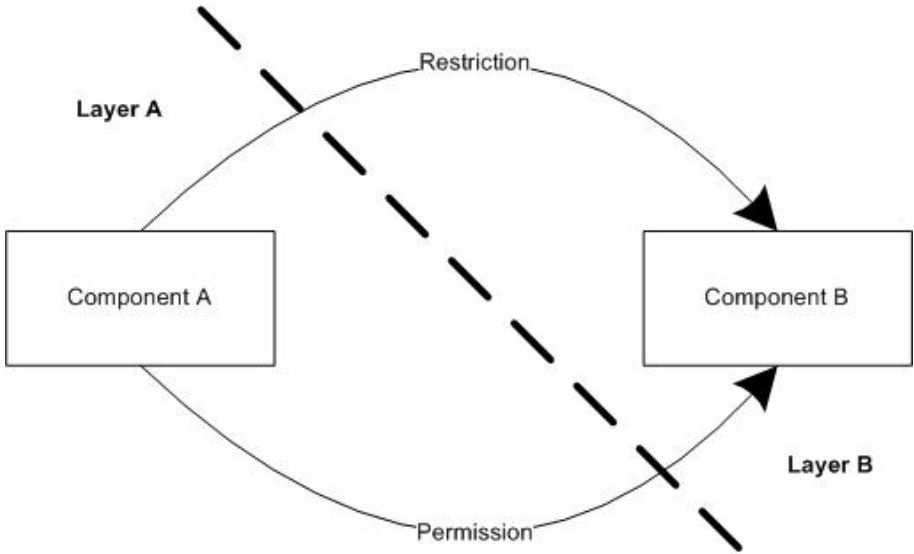


Fig. 1. Component restriction and permission

3 Component Properties

The components in this model of electronic voting schemes have a number of properties that can be defined much easier than similar properties for the system as a whole. The properties that distinguish one component from another include:

3.1 Layer and Slot into Which It Fits

The hierarchy of components is determined before the model is set up for a particular electronic voting system, and logically declaring the different parts of “a” system before filling in those slots with a particular implementation does seem to restrict the developer in some sense but on the other hand it does result in a system that can be more easily explained and developed. From a verifiability perspective such a system could also have associated with each component slot methods for checking that those components are correct.

3.2 Origin, Requirements and Constraints Posed by the Slot

This set of properties describe the requirements on the component which fits into the slot. As the requirements come from the model, rather than being made up on the fly by the developer of the component, it is possible to change between two components.

3.3 Verification Strategy

In our opinion the electronic voting system research community can no longer concern itself with systems that are not fully verifiable. In fact, we should reserve the term “electronic voting system” for something that is transparent and verifiable. Thus the requirement on every single component of the component based electronic voting system is that there exists some verification strategy — again, in our opinion, a combination of voter verifiability and public verifiability should be able to cover all components.

This may be one of the most attractive aspects of the component based methodology. By providing some overview of the verifiability of each component the developer can show that the full system is verifiable. By slotting one component out and another in, the same verification strategy can still be in place, reducing the work needed to compose a new component for a particular slot.

As an example of this we can mention the anonymisation strategy component in the computational layer. Let us say that a decryption mix network makes up this strategy (using the cryptography defined in the cryptography scheme component). The verification strategy on this component is a variant of zero knowledge proof. If a developer slots this anonymisation strategy component out and replaces it with a re-encryption mix network instead, the verification strategy remains the same.

3.4 Location in the System (Authority-Close or Voter-Close)

A component has a particular position in the system as a whole and this may be described by whether it is close to the voter and controlled by her or if it is close to an election authority and controlled by them.

One example of this might be the ballot form component in different versions of Prêt à Voter. In Prêt à Voter 2005 [1] the ballot form is quite authority-close in that it is set up by an election authority before the election and this authority holds all information needed to decrypt the receipt. To combat the chain voting attack [6] Prêt à Voter 2006 [7] introduces re-encryption mixes whereby the creation of the decryption information is distributed among a number of trusted parties. The form is printed on demand by the voter in the booth and can thus be regarded to be voter-close.

3.5 Threat Analysis

Each component, just as it has a verification strategy, can have a threat analysis attached to it. This analysis holds for any component that is slotted into the same place. The output of such a threat analysis is a number of requirements on the component.

3.6 Computational Complexity Analysis

Seemingly more dependent on the particular component, this property is part of the computational complexity analysis of the complete system. By decomposing

the system into smaller parts this analysis also becomes much easier, aiding the implementation onto hardware.

4 Examples

In this section we apply this model to two already existing schemes, in the first instance Punchscan [4,2,3] and in the second Prêt à Voter [1,5,6,7]. These simplistic decompositions are included to illustrate that the model would be applicable to schemes that already exist and that the further development of those schemes could be done in a component based methodology.

4.1 Punchscan Decomposed

Punchscan relies heavily on a central election authority to set up, manage and guarantee the safety, security and reliability of the voting system. The anonymisation strategy can be audited using a zero knowledge proof method.

A simple decomposition of Punchscan is done here. A Punchscan developer can of course make the decomposition much, much more detailed — at which point it becomes useful. When the decomposition has been made of course the continued development of the system is done in the component oriented methodology.

Human layer

Voter registration. The voter returns a form delivered to her house by local government. When she is registered a polling card is delivered by post.

Ballot form configuration. The ballot form is made up of two pages, the first has holes through which the second can be seen. On the first page is printed the races and the candidates in those races. Next to each candidate can be found a symbol which matches a symbol on the second page, visible through one of the holes. To vote the voter marks both pages using a bingo marker (“dauber”) over the second page symbol which corresponds to the candidate she wishes to vote for.

Polling station layout and management. When the voter enters the polling station her name is checked against the register. She identifies herself using the polling card.

Verifiability front-end. Voters can check their receipts on a web site as well as in the local media. The audit of the full election can be viewed online or through media reports.

Election layer

Election method. The local representative is elected by first past the post and thus each voter gives her vote to a single candidate.

Election management system. Before election day the election authority which oversees the election calls a meeting of election officials and representatives of the different political parties. In this meeting the election is set up on a diskless workstation running trusted, audited software. The output of this process is a database which is kept secret by the election authority for the purpose of decrypting votes after the election together with instructions to a print company which will print the ballot forms.

Voter-ballot box communication channel. The voter is allowed to fill out the ballot form in the privacy of a voting booth. Before leaving the booth she is able to create the encrypted receipt, making it hard for anyone to capture information from both layers of the ballot form. The encrypted receipt is scanned (with the aid of poll station workers) and transmitted electronically.

Computational layer

Cryptography scheme. Although no cryptography as such is used in Punchscan, the vote is hidden behind a number of random translations that are in turn captured in a number of interlinked *decryption tables*. There exists one translation for each of the two pages in each decryption table (the translation may be “no translation”) and a number of decryption tables may be linked together.

Anonymisation strategy. The decryption tables are set up in advance such that when a ballot form is made into an encrypted receipt, the application of all the translations for that form in all decryption tables reveals the voter’s intention. In order to anonymise a plaintext vote the full batch of encrypted receipts are secretly mixed after the application of each decryption table. The decryption is audited by making the authority commit to the decryption of a batch and then challenge it to reveal a number (but not all) links between the input and output batches. By not auditing all links the full link from an encrypted receipt through to a plaintext vote is broken at some stage.

Tallying procedure. The encrypted receipts are published to the web, the election authority applies its decryption tables and anonymisation strategy and publishes the result of each step. When the plaintext votes appear at the end the authority, and anyone wishing to check the result, can perform the tally.

Physical layer

Hardware authentication method. Each polling station machine is issued a cryptographic key pair and an identity.

Publishing strategy. All encrypted receipts received by the central repository are published, in some predetermined batch mode, to a publicly accessible read-only online resource.

Transfer method. Communication between the polling station and the central repository is encrypted using SSL.

4.2 Changes to Cryptography Component in Prêt à Voter

In the original Prêt à Voter [1] the cryptography scheme component is RSA, providing services to the anonymisation strategy component, which consisted of a decryption mix network. This system suffered from the authority knowledge (or chain voting) problem [6] and thus the next version [7] slotted in the RSA cryptography component and slotted in an Elgamal cryptography component. This new component was able to support a re-encryption mix network in the anonymisation strategy component and this in turn meant the the ballot form could be printed on demand in the booth, affecting the ballot form component.

4.3 Using Punchscan Style Ballot Form in Prêt à Voter

Quite interestingly we can show in this section that the “traditional” Prêt à Voter [1] ballot form is completely interchangeable with the “traditional” Punchscan [3] ballot form. The ballot form is thus a prime candidate for a component in the electronic voting system decomposition.

The ballot form in Prêt à Voter, shown in Figure 2, has a randomly ordered candidate list in the left of two columns. The right column consists of a grid where the voter marks her choice(s). Below this grid is printed the *onion* which encapsulates the order of the candidate list under a number of cryptographic layers. When the form is torn along a vertical perforation between the two columns and the left column shredded, the encrypted receipt remains. The onion value uniquely identifies the ballot form.

The Punchscan ballot form consists of two pages, shown in Figure 3, on the first of which is printed the candidate list in the canonical order. Next to each candidate on this page is also printed a symbol which corresponds to the same symbol shown, through holes in the first page, on the second page. The voter marks her choice using a bingo marker/dauber. This makes a mark on both pages at the same time and when one page is randomly selected and the other shredded, what remains is an encrypted receipt. The ballot form serial number printed in the top-right corner uniquely identifies the ballot form.

In both schemes the encrypted receipt is scanned and transmitted digitally. We can now describe both these forms with a component with the following configuration:

Layer and slot into which it fits. Human layer, Ballot form configuration slot.

Origin, requirements and constraints posed by the slot. The ballot form must list the candidates in the race for which it is valid on one half of the form. The other half must accept the voter’s intention. If the two halves are separated one remaining half must not reveal the candidate(s) for which the vote was cast but must be decryptable to reveal this information.

The form must be printed on paper, security paper is permitted. It must be scannable and shreddable.

LISA	
MAGGIE	
HOMER	
MARGE	
BART	

a45K1s

Fig. 2. The Prêt à Voter ballot form

82347

(c) Homer
(a) Marge
(b) Bart
(e) Lisa
(d) Maggie

d
b
e
a
c

Fig. 3. The Punchscan ballot form

Verification strategy. A reference printed on the form must uniquely identify it so that the voter may search for it in an online resource after the close of the election. During the election the voter may also use this reference to audit a form which will not be used for voting.

Location in the system (authority-close or voter-close). The ballot form is created in advance by an election authority, distributed under guard to safeguard the secret of the form and picked out at random by the voter.

Threat analysis. The ballot form must never be seen by anyone other than the voter before one half is removed to form the encrypted receipt as this may remove the system’s coercion resistance [6]. The ballot form is therefore to be distributed within an envelope that some physical procedure must guarantee that only the voter may open within the booth.

Computational complexity analysis. Creating the ballot form involves creating some decryption information, applying it a number of times over and then printing the form.

5 Discussion

This section provides a quick overview of some of the caveats with the component based model that we have foreseen.

5.1 Impossible to Have Strict Boundaries between Components

Defining electronic voting schemes as component based systems provides researchers with the opportunity to focus development on a particular component or to compare two different components that fit the same slot to determine which is best. However, when the component based model is turned into an implementation there may be complications. If an implementation is made of one particular component based model and one or more components are changed in the model then cascading these changes to the implementation may not be trivial. For example, the implementation of the mixing strategy may be heavily dependent on the cryptography scheme used and a change of the latter most likely results in the change of code in the earlier.

5.2 Restrictions on the Developer

Some may wish to make the developer of electronic voting systems completely without bounds — but this also implies that the developer will have no framework and the structure of the system developed will not be easily decomposed by others who wish to further the development.

5.3 Requirements Must Come from the Model

A developer of a component may look at the technical contribution of that component and make up the requirements of the component from that. This is easy to do but then suddenly the resulting system does not contain components that may be re-used or changed easily. Therefore it is important that the electronic voting system developer looks at the system as a whole and fully defines the requirements on each component before proceeding to create those components.

6 Summary

We have presented a first overview of a component based methodology for developing electronic voting systems. By not changing the complete system we hope that developers may be encouraged to look in depth on a particular component or set of components, providing a complete threat analysis as well as verification strategy for each.

Acknowledgements

Many thanks to the WOTE reviewers for their comments and to Zhe Xia, Steve Schneider, Peter Ryan, James Heather and Roger Peel for interesting conversations.

References

1. Chaum, D., Ryan, P.Y.A., Schneider, S.: A practical voter-verifiable election scheme. In: di Vimercati, S.d.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 118–139. Springer, Heidelberg (2005)
2. Fisher, K., Carback, R., Sherman, T.: Punchscan: Introduction and system definition of a high-integrity election system. In: Pre-Proceedings, IAVoSS Workshop on Trustworthy Elections, pp. 19–29 (2006)
3. Popoveniuc, S., Hosp, B.: An introduction to punchscan. In: Pre-Proceedings, IAVoSS Workshop On Trustworthy Elections, pp. 27–34 (2006)
4. Punchscan. Punchscan website, <http://www.punchscan.org>
5. Ryan, P.: A variant of the chaum voter-verifiable scheme. In: Proceedings of the 2005 Workshop on Issues in the Theory of Security, pp. 81–88 (2005)
6. Ryan, P., Peacock, T.: Prêt à voter: a system perspective. Technical Report of University of Newcastle, CS-TR:929 (2005)
7. Ryan, P.Y.A., Schneider, S.A.: Prêt à voter with re-encryption mixes. In: Gollmann, D., Meier, J., Sabelfeld, A. (eds.) ESORICS 2006. LNCS, vol. 4189, pp. 313–326. Springer, Heidelberg (2006)

A Verifiable Voting Protocol Based on Farnel (Extended Abstract)

Roberto Araújo¹, Ricardo Felipe Custódio², and Jeroen van de Graaf³

¹ TU-Darmstadt, Hochschulstrasse 10, 64289 Darmstadt - Germany
`rsa@cdc.informatik.tu-darmstadt.de`

² UFSC, Campus Universitário, Trindade, 88040-900 Florianópolis (SC) - Brazil
`custodio@inf.ufsc.br`

³ UFMG, Av. Antônio Carlos 6627, 31270-901 Belo Horizonte (MG) - Brazil
`jvdg@ufmg.br`

Abstract. Farnel is a voting system proposed in 2001 in which each voter signs a ballot. It uses two ballot boxes to avoid the association between a voter and a vote. In this paper we first point out a flaw in the ThreeBallot system proposed by Rivest that seems to have gone unnoticed so far: it reveals statistical information about who is winning the election. Then, trying to resolve this and other flaws, we present a new, voter-verifiable version of the Farnel voting system in which voters retain copies of ballot IDs as receipts.

Keywords: security, voting protocols, voter verification, paper-based, Farnel.

1 Introduction

Secure voting systems are a cornerstone of modern democratic societies. They can prevent or detect frauds or faults, and so provide accurate results. To increase transparency in such systems, researchers have been designing voter-verifiable schemes. These schemes allow the voter to verify whether her vote was taken into account in the result, but without violating the vote privacy.

Different strategies have been used to design voter-verifiable schemes. Almost all solutions described in the literature uses cryptography as basis, but the resulting protocols are often hard to grasp by a common person. Recently, a new kind of scheme with verification property was proposed by Rivest [10] - the ThreeBallot voting system. His proposal attempts to satisfy the voter verifiability *without* employing cryptography. Many drawbacks, though, have been reported for this scheme and improvements were incorporated in its newer versions.

In 2001, Custódio [3],[4] proposed a protocol, called Farnel [1], in which uses two ballot boxes and the voters sign ballots. In fact, Rivest uses the concept of the Farnel to sidestep a flaw in his scheme.

This paper presents a new version of Farnel, which is voter-verifiable. Also, it points out another flaw in the ThreeBallot scheme which seems to have gone

¹ Farnel means basket in Portuguese.

unnoticed so far; it leaks information. We do this as follows: Section 2 describes the original Farnel protocol. Section 3 shows how the ThreeBallot protocol leaks information. Section 4 describes the new Farnel protocol; it inherits some interesting characteristics that can be incorporated to obtain a verifiable voting system. Section 5 presents an electronic version of our protocol. Finally, Section 6 presents our conclusions.

2 The Original Farnel Scheme

Farnel [3,4] was conceived to address the problems of a conventional ballot box. This paper-based scheme requires each voter to sign one ballot. However, in order to avoid an association between the voter and her ballot, the voter does not sign her *own* ballot, but another one chosen at random, as explained below. This way it is possible to know who the voters were, and any attempt to add, modify, or delete votes, after the voting period, can be detected.

Initialization phase. Farnel uses *two* ballot boxes. Before voting starts, the first ballot box is publicly initialized with ballots filled out and signed by a ballot authority. This set of ballots must represent, with an equal probability, all possible votes. The second ballot box starts empty.

Voting phase. In order to vote, the voter receives a blank valid ballot (signed by the ballot authority), makes her choice, and casts the ballot into the first (pre-initialized) ballot box. Then, through manual or mechanical shuffling, the first ballot box presents a ballot chosen randomly from its current set of votes to the voter. After receiving the ballot, the voter signs and drops it into the second box. This ends the voting process for the voter.

Tallying phase. After the voting period has finished, the ballot authority opens and signs a second time all the votes of the first ballot box and adds them into the second box. Then the second box is opened and all ballots are counted. From this result the ballots from the initialization step are subtracted.

Properties of Farnel. Farnel gives warranties to the voter that her ballot will be counted, and that the exclusion or the addition of new votes is not possible after the voting phase. Anyone can, for example, verify that all ballots are signed, either by the voters or by the precinct. Moreover, everybody can check who voted without needing the list of voters. The scheme, however, is not voter-verifiable.

3 Information Leakage in the ThreeBallot Voting System

We give a brief description of Rivest's ThreeBallot voting scheme [10]. It gets its name from the fact that each ballot consists of three columns, each representing a full ballot. Each row of the ballot has a candidate name, and a ballot must have exactly one of the three cells following the candidate name marked. However,

candidate 1	X	X
candidate 2		X
candidate 3		X

Fig. 1. A ballot for candidate 1

the candidate that gets chosen will have two cells marked. For instance, in the ballot of Figure 1 the voter chose candidate 1.

Then the ballot is cut vertically, separating the three columns. One of the columns is copied; this is the voter's receipt. All three columns end up in the ballot box.

When the voting phase has completed, all votes are tallied. Obviously each candidate gets one free vote per ballot, so these votes must be subtracted to obtain the final tally.

There is a flaw with this scheme which is not mentioned in the latest version dated October 1, 2006; information about the contents of the ballot box is leaking before the election has finished.

When reading the ThreeBallot paper superficially it may appear that the secrecy of the ballot is perfect, i.e., that no information leaks. However, each receipt in fact does reveal a tiny bit of information, so little that it cannot be used against the voter. But in a large set of receipts statistical pattern do emerge.

The issue is best explained using an extreme example: suppose that candidate 1 gets all the votes and the other two none (we are assuming 3 candidates). Furthermore, suppose that all voters behave uniformly random with regard to where they put the marks and which column they choose as a receipt. Finally, suppose that all voters are willing to show their receipt to some organization who are at the polling station awaiting people who have just voted.

Counting the number of marks for each candidate (row) on the receipts reveals information on who is winning the election at that particular polling place. In this example, the winning candidate can expect $2/3$ mark per receipt, whereas all the others can expect only $1/3$ mark per receipt. The information is of a statistical nature.

To show the effect we wrote a small simulation program. Table 1 shows ten simulations for an election with three candidates, where 100 receipts have been collected and candidate 1 gets all the votes. The lines show the number of marks for each candidate, leaving no doubt at all about who is winning already while voting is still going on.

In fact we are dealing with two (p, n) -Bernoulli distributions: one with $p = 2/3$, and the others with $p = 1/3$. In both cases $n = \#$ receipts.

Observe that adding candidates (rows) to the ballot does not help. Adding columns does, because it flattens the distributions ($p = 1/4$ vs. $p = 2/4$; $p = 1/5$ vs. $p = 2/5$ etc.), but this is undesirable for practical reasons.

Observe also that a statistical analysis is more difficult if the voters do not behave randomly and the original scheme is used: the voter chooses which column to copy.

Table 1. A simulation of ten elections where every voter votes for candidate 1 and 100 receipts are collected

1	2	3	4	5	6	7	8	9	10
69	73	61	65	65	64	65	65	68	61
34	39	32	37	29	32	30	31	29	34
43	34	31	37	30	37	37	28	26	27

The flaw in the ThreeBallot system is debatable. It is true that the information obtained from the receipts has the same effect as exit polls. But there is a difference: not every country has or allows exit polls, and in addition voters can lie about how they voted, whereas in the threeballot system the receipts reveal actual information. In an election where the difference of votes among two candidates is small, for example, the information obtained from the receipts can certainly influence voters while the election is going on.

4 A Variant of the Farnel Scheme

As presented in Section 2, the Farnel scheme is not voter-verifiable; it just ensures, through signatures, that after the voting phase votes cannot be excluded and that new votes cannot be added. In this Section we present a new paper-based scheme inspired on Farnel. It also uses two ballot boxes, but does not depend on signatures. It provides a receipt to the voter, but without leaking information during the voting phase.

4.1 Prerequisites

The ballot form. The ballot form used is composed of two halves. The first half is not much different from the layout traditionally used in elections. It is composed of a list of voting options (including a blank vote option) where next to each option there is a space to select it. It also contains an identification number (ID) which identifies the ballot uniquely and associates it to the election. The second half contains only the same ID (see Section 4.4 for a discussion about the IDs). The halves are separated by a perforation to allow detachment by the voter and the IDs are covered by scratch surfaces (see also Figure 2).

The ballot boxes. Two ballot boxes are used. One of them is a conventional box; however, it must be initialized with filled out, fake ballots (i.e. just the part that contains the options) before the voting starts (otherwise the first voters would not have a set of random IDs to choose from).

The other ballot box is able to receive a slip containing an ID, to add it to a set of already received slips, and to copy l randomly chosen IDs from this set. To this end we assume that the box has some mechanism to shuffle its contents, and that copies are made in a memoryless way. The shuffling mechanism, for example, could be based on a bingo cage. For convenience, we call this special ballot box *Farnel* in the text.

4.2 The Protocol

Initialization phase. In this phase the ballot authority establishes the following *voting parameters*: a number x of initial votes and a number l of IDs that should be printed on the receipts (see Section 4.4 for a discussion about these parameters). Moreover, he initializes publicly the ballot boxes. Let's say that there are v eligible voters in the election.

Before initializing the ballot boxes, the ballot authority performs a cut-and-choose process to prove the correct formation of the ballots. For a number $2x$ of blank ballots, he takes x ballots at random, detaches their protective layers, and publishes them; these ballots are no longer used. After that, the authority holds the other (entire) x blank ballots and tears each of them in two along the perforation. Next, he marks an option on each of the parts containing the options, detaches their layers, and casts them into the conventional ballot box. The options can be selected at random, but each of them should have at least one vote. The authority then scratches away the layers of the other parts (the slips that contain copies of the IDs) and casts them into the Farnel ballot box. The total number of fake votes for each option is published. Finally, the authority seals both boxes until the voting begins. **Note that neither the authority nor third parties should be able to record or remember the IDs of the initial votes.**

Voting phase. After proving her eligibility to the voting authorities, the voter receives a blank ballot form. The following steps are performed to cast a vote and to obtain the receipt (see also Figure 2).

1. (Verifying and filling out the ballot form) The voter scratches away the layer covering the IDs and matches them (a). If they are equal, she makes her vote by marking one of the options available (b). We assume that the voter cannot record or remember the ballot's ID.
2. (Casting the vote) The voter separates the two parts (c) of the ballot form, casting the part containing the ballot ID and the options into the conventional ballot box (d). The other part, showing only the ID, is cast into the Farnel ballot box (e).
3. (Obtaining the receipt) The Farnel ballot box is shuffled (f) and l copies are produced of IDs which are printed as a receipt to the voter (g).

Tallying phase. In a public session the talliers open the two ballot boxes and publish their contents on the bulletin board. To compute the results of the election, all votes are tallied. The fake ballots cast in the initialization phase are subtracted from the sums yielding the final result.

Ballot verification. Anyone can check on the bulletin board whether each ballot from the conventional ballot box has a corresponding ID in the Farnel ballot box. In addition, the voters confirm whether their receipts (i.e. the IDs) match to ballots on the bulletin board. If one ballot and its ID were not published, the voter can complain by showing her receipt to a voting authority.

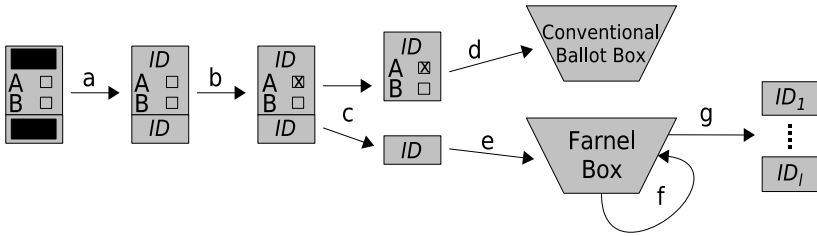


Fig. 2. Main voting steps of the new paper-based scheme

4.3 Security Requirements and the New Paper-Based Scheme

Here we sketch an analysis of our scheme based on the security requirements normally found in the literature. In this analysis, we supposed that the bulletin board cannot be compromised.

Accuracy. In our scheme duplication, elimination, substitution, and addition of votes can be detected. The detection is accomplished by checking the information published on the bulletin board. Duplicates can be identified by checking if the IDs of the votes published are unique. Anyone can also detect elimination and substitution of votes. Every vote on the board should have a corresponding ID published. Moreover, the voters can independently match their receipts (i.e. the IDs) to the votes on the board. Note, though, that the detection is probabilistic since not all votes will have their IDs printed on the receipts. The addition of votes can be detected through the total number of votes published. The total should be the sum of the number of initial votes and of the number of voters that cast their votes.

Privacy. The voter privacy in our scheme is ensured even if she wants to violate it as follows. An adversary could try to violate the privacy by: obtaining an ID of a specific vote or extracting information about votes from the receipts.

In the first case, a voter or a voting authority could attempt to remember or record the ID of a ballot. In order to prevent this, the ballots in our scheme have their IDs covered by scratch surfaces. We suppose, though, that the voter cannot remember or record the ID of her own ballot (see Section 4.4 for a discussion about the IDs).

In the second case, an adversary could ask a voter to point the ID of her vote on her receipt. As the receipt is composed of a set of IDs chosen at random, the voter can only try to guess an ID related to her option. Again, we consider that the voter cannot remember the ID of her own ballot.

Alternatively, the adversary could collect the receipts of most of the voters and try to determine the votes of the first voters; we call this *attack of collecting receipts*. He could explore the fact that the IDs of the first voters are more probable to appear on the receipts than the IDs of the last ones. To attempt to determine the votes, the adversary would check the most repeated IDs on the receipts and match them later to the votes on the board. Note, though, that the

IDs cast before the election are as probable as the IDs of the first voters and that the adversary cannot distinguish among them at least as long as there is one initial vote for each option.

Verifiability. Our scheme can be verified by voters and by third parties. The IDs on Farnel ballot box aim at verifying the votes on the conventional ballot box. The publication of the IDs and of the votes allows anyone to verify the exactness of the voting results.

The voter verifiability in our proposal is different from the normally found in the literature. Instead of verifying if her own vote is in the final tally, the voter verifies a small subset of all votes. This is accomplished by matching the IDs on the receipt to the votes on the board. Note that here the verifiability depends on the *voting parameters*.

Voter-verifiable election schemes usually take into account that the voters will use their receipts to verify their ballots. Karlof et al. [7], though, pointed out that some voters can discard their receipts. Consequently, an adversary could take advantage of this and replace votes without being detected. Our scheme employs two approaches to mitigate this problem. First, each receipt is composed of a set of IDs and these IDs (or some of them) can be printed in other receipts. This way, even if a voter discards her IDs, other voters can possibly verify the votes related to these IDs. Second, the Farnel ballot box maintains the IDs of all votes and they are also published. This way, it adds redundancy to the verification process.

4.4 Discussion

We discuss now some aspects inherent to our scheme.

The IDs. The IDs on the ballot form should be easy to compare and difficult to remember. The voter should compare the IDs to detect a possible malformation of her ballot (i.e. different IDs) and should not be able to remember it afterwards. Although these properties seem to be contradictory and difficult to implement, barcodes could be used to encode IDs and prevent the voters to recall them; the voter could compare barcodes easily as long as they are thick enough. However, some voters may not perform the comparison or ignore the malformation of their ballots.

A better solution is to avoid the voter comparing the IDs of her own ballot. This way, the ID could be just an alphanumeric string. The drawback is that malformed ballots would not be detected. To mitigate this, we employ a cut-and-choose process for auditing the ballots. That is, before receiving a blank ballot, the voter chooses some random ballots and detaches their scratches to verify their IDs; these ballots are discarded. As the voter does not remove the scratches of her own ballot, the Farnel ballot box now needs a special mechanism to remove the scratch of the slip; the other scratch can be removed in the tallying phase.



Fig. 3. Ballot form to prevent chain voting

Chain voting. This is a real threat to our scheme. An adversary can smuggle a valid blank ballot, mark an option on it, and corrupt a voter to use this ballot. After voting, the voter returns to the adversary the blank ballot received from the voting authorities. The adversary now can use the new blank ballot to corrupt another voter in the same way. Besides the usual chain voting attack, the adversary here can perform differently. He can obtain a slip containing an ID and corrupt a voter to use it; the voter gives back the adversary the ID of her own ballot and the adversary uses this ID to confirm the voter vote in the tallying phase. In both cases, therefore, the security of the scheme would be compromised.

In order to prevent these attacks, we modify our ballot form. We add a serial number to the ballot such as Jones [6]. The number is printed over the scratch surface that covers the ID on the slip. We also change the position of the other ID to allow the ballot to be folded showing just the scratch surfaces; now it is printed on the back of the ballot (see Figure 3 for an example of this ballot form). In addition to the ballot form, the following voting steps of the scheme must be modified. The authorities should record the serial number of the ballot before giving it to the voter and should confirm the number before the voter casts her vote. Note that the scratch containing the serial number should be removed after the authorities confirm it and that the voter does not compare the IDs on her ballot. As above, we assume that the scratch is removed by the Farnel ballot box.

The voting parameters. As described before, the voting parameters are composed of the number of initial votes x and the number of IDs l printed on the receipts. These parameters as well as the number of voters v affect the voter verifiability. Also, they are related to the privacy, that is, they can facilitate or not the attack of collecting receipts (see Section 4.3). From these remarks, we deduce the following:

Considering x much greater than the number of voters v (e.g. 10 times greater), the IDs cast by the voters will be almost statistically indistinguishable from the initial IDs. As result, if l is small (e.g. $l = 1$), an adversary cannot violate the voters privacy (particularly the privacy of the first voters) by distinguishing among the voters IDs and the initial IDs from the receipts. A small l , though, affects the voter verifiability as the chance of detecting problems in the tally decreases.

A v greater than x , on the other hand, results in more IDs of voters on the receipts even if l is small. Consequently, an adversary has more chance to distinguish among the voters IDs and the initial IDs. For example, if $v = 500$, $x = 2$, and $l = 4$, the voters IDs will appear more on the receipts than the initial ones and this facilitates the distinction.

Table 2. x - number of initial votes; l - number of IDs on each receipt; *1st ID* - prob. of the 1st ID to appear on at least one of the receipts; *Detection* - prob. of detecting an elimination of a vote in the tally by means of the receipts

x	100		300		1000			1500		
l	1	2	1	2	3	4	5	4	5	6
<i>1st ID</i> %	83	97	62	86	70	80	87	68	76	82
<i>Detection</i> %	49	66	43	63	60	69	76	61	68	74

Certainly, voter verifiability and privacy in our scheme are related. To measure the relation of these properties, we performed experiments considering a fix number of voters $v = 500$ for different l and x . The experiments show the probability of the ID cast by the first voter to appear on at least one of the 500 receipts. Also, they show the chance of detecting the elimination of a vote in the tally through the receipts. Table 2 presents some results of these experiments.

The last votes and the receipts. As described in Section 4.3, the initial IDs as well as the IDs cast by the first voters have more chance to appear on the receipts. This way, the votes corresponding to these IDs are more probable to be verified by the voters. Conversely, the IDs cast by the last voters have less chance to be printed on the receipts. That is, the probability of an ID₂ cast after ID₁ to appear on at least one receipt is less than the probability of ID₁ to appear on the receipts. Thus, considering a number of voters v , and the first and the last voter, we observe that the ID of the first voter can appear in any of the v receipts, while the ID of the last voter can appear only in the last receipt.

As the chance of the last IDs to appear on the receipts is less than the chance of first IDs, an adversary could substitute a vote of one of the last voters without being detected. Though, the adversary would need to identify these votes before substituting them. In principle, because the adversary cannot distinguish the first votes from the last ones, he cannot identify the last votes to replace them. Hence, the best that the adversary can do is to try to guess the last votes.

On the other hand, by observing most of the receipts, the adversary could identify the votes that would be possibly verified and substitute only the votes in which the IDs do not appear on the receipts. We call this *attack of substitution of votes without receipts*. As the Farnel box keeps all the IDs that compose the receipts and these IDs are published on the bulletin board immediately after the voting, the attack would be detected. However, the adversary can still succeed if he is able to substitute votes (without receipts) and their corresponding IDs before they are published.

In order to mitigate the attack, all IDs in the Farnel box should appear on the receipts. Although this could be achieved by increasing the number of IDs on the receipts, the IDs of the last voters still would have less chance to appear on them. As solution, we propose to print a set of receipts at end of the voting. In other words, after the last voter casts her ID and obtains her receipt, the Farnel box shuffles its IDs (without receiving an input), prints a predefined number of receipts, and outputs them. These receipts would be handed to help

organizations and could be also published on the bulletin board. Alternatively, the receipts could also be printed during the voting. However, the period defined for printing should consider the chance of the last votes to appear on the receipts.

Supervising the votes. The ballot design is fundamental for the verification of votes in our scheme. As presented, it is composed of two halves that have the same ID. The halves are separated (i.e. by casting the two halves into different boxes) during the voting phase and in the tallying phase they are associated (i.e. by publishing the halves) to allow anyone to verify the vote. Still, the ID printed on the halves may appear on different receipts, so the voters can verify the corresponding vote independently.

However, due to the verification through the IDs, the talliers must be trustworthy. Particularly, from the opening of the conventional ballot box until the publication of all votes on the bulletin board, the talliers should supervise strictly the votes. Otherwise, an adversary (e.g. a malicious tallier) may replace votes without being detected.

Suppose an adversary has access to the set of votes of the conventional ballot box before the votes are published on the bulletin board. In order to replace a vote, the adversary smuggles a vote from the set, makes a fake vote for a different option but using the same ID of the vote smuggled, and includes the fake vote in the set. This way, after publishing all ballots, the fake vote would appear on the board instead of the smuggled one.

This attack would undermine the security of the scheme. Because votes are verified through the IDs, voters and third parties would not detect the replacement of the original vote by the fake one. However, the attack can be defeated by using a ballot design in which the receipt part includes some information about the option selected. We leave this ballot design as future work.

5 An Electronic Version of the Paper-Based Scheme

We now introduce an electronic version of the scheme presented in the previous Section. It uses commitments as the IDs, which are constructed by a voting machine (DRE). Also, it uses a special ballot box which accepts a ballot ID and hands out copies of other ballot IDs.

5.1 Building Blocks

Threshold El Gamal Cryptosystem. As a basis for the scheme we employ the El Gamal public key cryptosystem [5] under a subgroup of order q of Z_p^* , where p and q are large primes and $q|p-1$. More specifically, we require a threshold variant, as described by Cramer et al. [2].

We use the following notation: T is an El Gamal public key corresponding to a secret key \hat{T} , while $E_T(i, s)$ is the El Gamal encryption of a message i constructed with T and a random number $s \in_R Z_q$, and $D_{\hat{T}}(i)$ is the El Gamal decryption of i .

Mix Net. In order to make messages anonymous during the tallying phase, we employ a mix net. This primitive was introduced by Chaum [1] and further improved by many others authors. Specifically, we require a verifiable reencryption mix net such as the proposal of Neff [8].

Commitment scheme. Another cryptographic primitive is a commitment scheme, which must be homomorphic and will be used to commit to the voting options. We use the El Gamal cryptosystem for this purpose. Though, the Pedersen commitment [9] could be also employed.

Cut-and-choose. We employ a cut-and-choose process to prove the voter that her vote was correctly formed by the voting machine. This is accomplished in a similar way to Lee et al. [11]. Especially, the voting machine makes a number of El Gamal ciphertexts and presents them to the voter; she then selects some ciphertexts at random for verification and the machine opens them by revealing the random numbers employed.

5.2 Prerequisites

The ballot. The ballot is constructed by the voting machine and is presented to the voter. It contains each possible option and some encrypted stuff next to it: each option i is associated to two commitments, as follows: $\langle i, \text{commit}(i, r_{i1}), \text{commit}(i, r_{i2}) \rangle$ for $r_{i2}, r_{i1} \in_R Z_q$. In particular, each commitment is represented by: $\text{commit}(i, r) = \langle E_T(i, s_1), E_T(r, s_2), E_M(ir, s_3) \rangle$ for $r, s_1, s_2, s_3 \in_R Z_q$. Here r is chosen uniformly at random from Z_q , and ir is the product of i and r , while (T, \hat{T}) stands for the El Gamal keys of the talliers, and (M, \hat{M}) stands for the keys of the special ballot box which uses the *same* El Gamal modulus.

The special ballot box. The paper-based scheme from the previous Section requires a ballot box that receives a ballot ID, shuffles its contents, and output copies of randomly chosen IDs. Here our special ballot box is initialized with a set of encrypted IDs which it keeps in a private list, L . It receives an enciphered ID, adds it to L , decrypts some random elements from L , and prints the result on the receipt. Elements selected are not deleted from the list, though.

There are four parties involved in our scheme:

Voters. The voters cast votes and receive receipts for checking data later. Each receipt is composed of three parts: the ballot (with commitments and hidden commitments) and some decommitments, the commitment of the option chosen, and some plaintexts from the list L .

Voting machine (DRE). The voting machine generates ballots, makes the first two parts of the receipts, and publishes commitments on the bulletin board.

Special ballot box. It holds a private list of encryptions L and acts as described before. It has a barcode reader and receives new ciphertexts through this reader. It also prints the last part of the receipt.

Tallying authorities. These authorities are responsible for running a mix net, and for decrypting and counting the votes. They also define the number of initial votes and generates them; in addition, they define the number of votes that each voter verifies. They hold the keys T, \tilde{T} . We suppose that a subset of the talliers are trustworthy.

5.3 The Protocol

Initialization phase. In this phase, the following parameters of the voting are established and published on the bulletin board: the voting options (or candidates), the number of initial votes, the number of IDs that should be printed on the receipt. Let's say that there are m options i ($i = 1, \dots, m$), a number x of initial votes, and a number l of IDs printed on the receipt.

The talliers generate the initial votes according to x and to the commitment scheme explained before. Then the talliers publish commitments of the form: $commit(i, r) = \langle E_T(i, s_1), E_T(r, s_2), E_M(ir, s_3) \rangle$ for different $r, s_1, s_2, s_3 \in_R Z_q$ in each commitment. The values $E_M(ir, s_3)$ are handled by the special ballot box as its private list L of encryptions.

Voting phase. After proving her eligibility to the voting authorities, the voter is allowed to interact with the DRE in the voting booth. The following steps are executed to cast a vote and to obtain the receipt. Figures 4 and 5 exemplify the scheme for three voting options.

1. (Generating the ballot). For each option i ($i = 1, \dots, n$), the DRE generates the triple: $\langle i, commit(i, r_{i1}), commit(i, r_{i2}) \rangle$ for $r_{i2}, r_{i1} \in_R Z_q$. As described before, each commitment is composed of: $\langle E_T(i, s_1), E_T(r, s_2), E_M(ir, s_3) \rangle$ for $r, s_1, s_2, s_3 \in_R Z_q$. After that, the DRE prints the ballot on the receipt (a). Here as well as in the next two steps the ballot is not shown to the voter.
2. (Opening some commitments). The voter informs the DRE to open the first or the second commitment for each option i . After this, the DRE opens the corresponding commitments (b). In other words, for a commitment $\langle E_T(i, s_1), E_T(r, s_2), E_M(ir, s_3) \rangle$ already printed on the receipt, the DRE prints r, s_1, s_2, s_3 on the receipt as decommitment.
3. (Voting). In order to vote, the voter informs her option i and the DRE prints the corresponding, not opened, commitment on the receipt (c). In particular, the DRE prints $\langle E_T(i, s_1), E_T(r, s_2), E_M(ir, s_3) \rangle$.
4. (Verifying the ballot). Now, the DRE shows the receipt (A) to the voter. The voter should verify if the commitments selected were opened and if the commitment corresponding to her vote was printed. If the receipt is correct, the voter confirms her vote. The DRE then prints a stripe on the not opened commitments of the ballot to erase them. She also prints the barcode of $E_M(ir, s_3)$ of the voter's vote and adds a digital signature to the receipt; the voter holds this receipt (B). The other elements of the vote, $\langle E_T(i, s_1), E_T(r, s_2) \rangle$, are sent to the bulletin board.

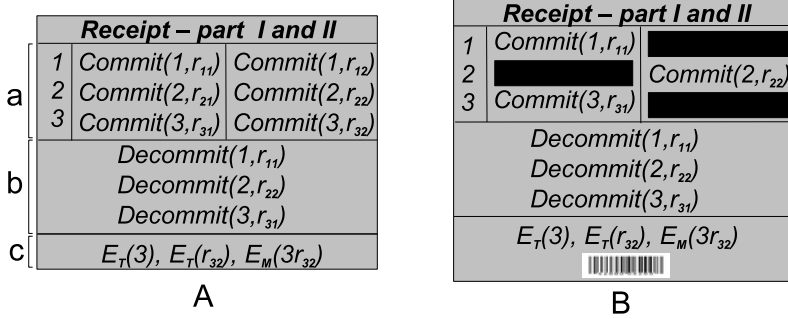


Fig. 4. Parts I and II of the receipt. A - the receipt that the voter verifies; B - the receipt that the voter holds.

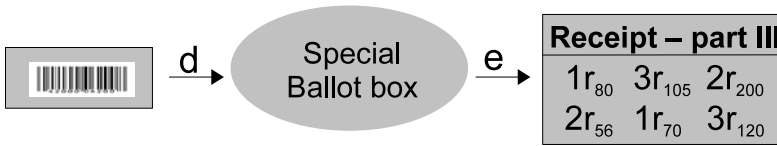


Fig. 5. The last part of the receipt

5. (Obtaining the last part of the receipt). Using the barcodes reader, the voter adds $E_M(ir, s_3)$ to the special ballot box (d). The box writes $E_M(ir, s_3)$ to its private list L . Then, it chooses y elements at random from L , decrypts them, and prints the results on the receipt (e). In doing so the elements are not deleted from L . Figure 5 illustrates this step.

Tallying phase. After the election, the talliers send all pairs of encryptions published on the bulletin board, $\langle E_T(i, s_1), E_T(r, s_2) \rangle$, to a mix net. The mix net shuffles the pairs and publishes them on the bulletin board. After this, the talliers cooperate to decrypt the pairs to obtain the options i and the random numbers r . The talliers then multiply i and r , and publish the triples $\langle i, r, ir \rangle$ on the bulletin board. To compute the elections results, all votes from the voting phase (i.e. the i 's) are counted and from this result are subtracted the fake votes generated in the setup phase.

Ballot verification. The voter receives a receipt composed of three parts. The first part, which contains the ballot (with commitments and hidden commitments) and the decommitments, is used to verify the construction of the ballot. This may be accomplished by a helper organization through a computer. It constructs the commitments from the decommitment values of the receipt and then checks if the resulting commitments match the commitments printed on the receipt.

The second part of the receipt, which contains the commitment of the option chosen, is used to check if the commitment of the receipt appears on the bulletin board.

The third part of the receipt contains a list of *ir* to verify if the values *ir* match the values published by the talliers on the board.

6 Conclusion

We have presented a flaw in the ThreeBallot voting system and a new version of the Farnel protocol which is voter-verifiable. Also, we have shown an electronic scheme based on the new proposal.

Our schemes introduce a new way to verify votes: the voter does not verify her own vote, but copies of a subset of votes cast so far. More precisely, the voter receives copies of some randomly selected ballot IDs. These are used later to compare with the IDs of the ballots published on the bulletin board.

The paper-based version relies on trustworthy talliers and on a special ballot box that can shuffle and copy receipts. Although trustworthy talliers may be a strong requisite, this requisite is not necessary as long as the receipts contain some information related to the options selected. Receipts with this property, though, depends on a new ballot design and are subject of future work.

We have used the paper-based version to model the electronic scheme. The proposal works as expected and (differently from the paper-based scheme) produces receipts connected with the options chosen, but it has several drawbacks. Especially, it requires a verifiable mix net in the tallying phase and the special ballot box must be reliable. We believe, though, that this scheme can be improved and are working in this direction.

References

1. Chaum, D.: Untraceable electronic mail, return addresses and digital pseudonyms. *Communications of the ACM* 24(2), 84–88 (1981)
2. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: Fumy, W. (ed.) *EUROCRYPT 1997*. LNCS, vol. 1233, pp. 103–118. Springer, Heidelberg (1997)
3. Custódio, R.: Farnel: um protocolo de votação papel com verificabilidade parcial. Invited Talk to Simpósio de Segurança em Informática (SSI) (November 2001)
4. Custódio, R., Devegili, A., Araújo, R.: Farnel: um protocolo de votação papel com verificabilidade parcial (2001) (unpublished notes)
5. El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakely, G.R., Chaum, D. (eds.) *CRYPTO 1984*. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)
6. Jones, D.W.: Chain voting (August 2005), <http://vote.nist.gov/threats>
7. Karlof, C., Sastry, N., Wagner, D.: Cryptographic voting protocols: A systems perspective. In: *Proceedings of the Fourteenth USENIX Security Symposium (USENIX Security 2005)*, August 2005, pp. 33–50 (2005)
8. Andrew Neff, C.: A verifiable secret shuffle and its application to e-voting. In: *ACM Conference on Computer and Communications Security*, pp. 116–125 (2001)

9. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
10. Rivest, R.L.: The threeballot voting system (October 2006), <http://people.csail.mit.edu/rivest/Rivest-TheThreeBallotVotingSystem.pdf>
11. Kim, S., Lee, Y., Lee, K., Won, D.: Efficient voter verifiable e-voting schemes with cryptographic receipts, June 2006. In: IAVoSS Workshop On Trustworthy Elections (WOTE 2006), Cambridge (2006)

Verifying Privacy-Type Properties of Electronic Voting Protocols: A Taster^{*}

Stéphanie Delaune¹, Steve Kremer¹, and Mark Ryan²

¹ LSV, ENS Cachan & CNRS & INRIA, France

² School of Computer Science, University of Birmingham, UK

Abstract. While electronic elections promise the possibility of convenient, efficient and secure facilities for recording and tallying votes, recent studies have highlighted inadequacies in implemented systems. These inadequacies provide additional motivation for applying formal methods to the validation of electronic voting protocols.

In this paper we report on some of our recent efforts in using the applied pi calculus to model and analyse properties of electronic elections. We particularly focus on anonymity properties, namely vote-privacy and receipt-freeness. These properties are expressed using observational equivalence and we show in accordance with intuition that receipt-freeness implies vote-privacy.

We illustrate our definitions on two electronic voting protocols from the literature. Ideally, these properties should hold even if the election officials are corrupt. However, protocols that were designed to satisfy privacy or receipt-freeness may not do so in the presence of corrupt officials. Our model and definitions allow us to specify and easily change which authorities are supposed to be trustworthy.

1 Introduction

Electronic voting protocols. Electronic voting promises the possibility of a convenient, efficient and secure facility for recording and tallying votes. It can be used for a variety of types of elections, from small committees or on-line communities through to full-scale national elections. Electronic voting protocols are formal protocols that specify the messages sent between the voters and administrators. Such protocols have been studied for several decades. They offer the possibility of abstract analysis of the voting system against formally-stated properties.

In this paper, we recall some existing protocols which have been developed over the last decades, and some of the security properties they are intended to satisfy. We focus on privacy-type properties. We present a framework for analysing those protocols and determining whether they satisfy the properties.

^{*} This work has been partly supported by the EPSRC projects EP/E029833, *Verifying Properties in Electronic Voting Protocols* and EP/E040829/1, *Verifying anonymity and privacy properties of security protocols*, the ARA SESUR project AVOTE and the ARTIST2 NoE.

Properties of electronic voting protocols. Some properties commonly sought for voting protocols are the following:

- *Eligibility*: Only legitimate voters can vote, and only once.
- *Fairness*: No early results can be obtained which could influence the remaining voters.
- *Individual verifiability*: A voter can verify that her vote was really counted.
- *Universal verifiability*: The published outcome really is the sum of all the votes.
- *Vote-privacy*: The fact that a particular voter voted in a particular way is not revealed to anyone.
- *Receipt-freeness*: A voter does not gain any information (a *receipt*) which can be used to prove to a coercer that she voted in a certain way.
- *Coercion-resistance*: A voter cannot cooperate with a coercer to prove to him that she voted in a certain way.

The last three of these are broadly *privacy-type* properties since they guarantee that the link between the voter and her vote is not revealed by the protocol. The weakest of the three, called *vote-privacy*, roughly states that the fact that a voter voted in a particular way is not revealed to anyone. *Receipt-freeness* says that the voter does not obtain any artefact (a “receipt”) which can be used later to prove to another party how she voted. Such a receipt may be intentional or unintentional on the part of the designer of the system. Unintentional receipts might include nonces or keys which the voter is given during the protocol. Receipt-freeness is a stronger property than privacy. Intuitively, privacy says that an attacker cannot discern how a voter votes from any information that the voter necessarily reveals during the course of the election. Receipt-freeness says the same thing even if the voter voluntarily reveals additional information.

Coercion-resistance is the third and strongest of the three privacy properties. Again, it says that the link between a voter and her vote cannot be established by an attacker, this time even if the voter cooperates with the attacker during the election process. Such cooperation can include giving to the attacker any data which she gets during the voting process, and using data which the attacker provides in return. When analysing coercion-resistance, we assume that the voter and the attacker can communicate and exchange data at any time during the election process. Coercion-resistance is intuitively stronger than receipt-freeness, since the attacker has more capabilities.

Verifying electronic voting protocols. Because security protocols in general are notoriously difficult to design and analyse, formal verification techniques are particularly important. In several cases, protocols which were thought to be correct for several years have, by means of formal verification techniques, been discovered to have major flaws. Our aim in this paper is to use and develop verification techniques, focusing on the three privacy-type properties mentioned above. We use *the applied pi calculus* as our basic modelling formalism [2], which has the advantages of being based on well-understood concepts. The applied pi calculus has a family of proof techniques which we can use, and it is partly

supported by the ProVerif tool [5]. Moreover, the applied pi calculus allows us to reason about equational theories in order to model the wide variety of cryptographic primitives often used in voting protocols.

As it is often done in protocol analysis, we assume the Dolev-Yao abstraction: cryptographic primitives are assumed to work perfectly, and the attacker controls the public channels. The attacker can see, intercept and insert messages on public channels, but can only encrypt, decrypt, sign messages or perform other cryptographic operations if he has the relevant key. In general, we assume that the attacker also controls the election officials, since the protocols we investigate are supposed to be resistant even if the officials are corrupt. Some of the protocols explicitly require a trusted device, such as a smart card; we do not assume that the attacker controls those devices.

Outline of the paper. In Section 2, we recall the basic ideas and concepts of the applied pi calculus. Next, in Section 3, we present the framework for formalising voting protocols from the literature, and in Section 4 we show how two of our three privacy-like properties are formalised, namely vote-privacy and receipt-freeness. (We omit the formalisation of coercion-resistance for reasons of space; see [9].) In Sections 5 and 6, we recall two voting protocols from the literature, and show how they can be formalised in our framework. We analyse which of the properties they satisfy. This paper summarises and provides a taster for a much longer paper currently being submitted to the *Journal of Computer Security* [9]. In this paper we intend to give the flavour of our work without going into great detail.

2 The Applied pi Calculus

The applied pi calculus [2] is a language for describing concurrent processes and their interactions. It is based on an earlier language called the pi calculus, which has enjoyed a lot of attention from computer scientists over the last decades because of its simplicity and mathematical elegance. The applied pi calculus is intended to be much richer than the pi calculus, while keeping its mathematical rigour, and is therefore more convenient to use in real applications. The applied pi calculus is similar to another pi calculus derivative called the spi calculus [3], but the spi calculus has a fixed set of primitives built-in (symmetric and public-key encryption), while the applied pi calculus allows one to define a wide class of primitives by means of an equational theory. This is useful in electronic voting protocols, where the cryptography is often sophisticated and purpose-built. The applied pi calculus has been used to study a variety of security protocols, such as a private authentication protocol [11] or a key establishment protocol [1].

2.1 Syntax and Informal Semantics

Messages. To describe processes in the applied pi calculus, one starts with a infinite set of *names* (which are used to name communication channels or other

atomic data), an infinite set of *variables*, and a *signature* Σ which consists of the *function symbols* which will be used to define *terms*. In the case of security protocols, typical function symbols will include `enc` for encryption, which takes plaintext and a key and returns the corresponding ciphertext, and `dec` for decryption, taking ciphertext and a key and returning the plaintext. Terms are defined as names, variables, and function symbols applied to other terms. Terms and function symbols are sorted, and of course function symbol application must respect sorts and arities. When the set of variables occurring in a term T is empty, we say that T is *ground*.

Example 1. Let $\Sigma = \{\text{enc}, \text{dec}\}$, where `enc` and `dec` are each of arity 2. Suppose a, b, c are names (perhaps representing some bitstring constants or keys), and x, y, z are variables. Then `enc(a, b)` is a ground term (which represents the encryption of a using the key b). The term `dec(enc(a, b), y)` is also a term (but not a ground term), representing the decryption by y of the result of encrypting a with b . The symbols `enc` and `dec` may be nested arbitrarily.

By the means of an equational theory E we describe the equations which hold on terms built from the signature. We denote $=_E$ the equivalence relation induced by E . Two terms are related by $=_E$ only if that fact can be derived from the equations in E .

Example 2. A typical example of an equational theory useful for cryptographic protocols is `dec(enc(x, y), y) = x`. In this equational theory, we have that the terms $T_1 := \text{dec}(\text{enc}(\text{enc}(n, k), k'), k')$ and $T_2 := \text{enc}(n, k)$ are equal, i.e. $T_1 =_E T_2$, while obviously the syntactic equality $T_1 = T_2$ does not hold.

Equational theories are the means by which we represent cryptographic operations. We do not model the mechanisms (whether bitstring manipulation or numerical calculation) that constitute the cryptographic operations. Rather, we model the behaviour they are designed to exhibit. Thus, stipulating the equation `dec(enc(x, y), y) = x` models symmetric encryption. In the model terms are unequal unless they can be proved equal by the equations. This means that the only way of recovering x from `enc(x, y)` is by the application of `dec(·, y)` (and in particular, the agent that makes that application is required to know the key y).

If M and N are terms, then the pair (M, N) is a term, and from it may be extracted the components M and N . Formally, this requires us to introduce the binary “pairing” function (\cdot, \cdot) and the projection functions proj_1 and proj_2 , but usually we don’t bother with that and keep the equational theory for pairs (and tuples of any finite length) implicit.

Processes. In order to model the dynamic part of protocols, we require processes. In applied pi, there are two kinds of processes, namely *plain processes*, denoted by P, Q, R and *extended processes*, denoted by A, B, C . In the grammar described below, M and N are terms, n is a name, x a variable and u is a metavariable, standing either for a name or a variable.

$P, Q, R :=$ plain processes	$A, B, C :=$ extended processes
0	P
$\text{in}(u, x).P$	$A \mid B$
$\text{out}(u, N).P$	$\nu n.A$
$\text{if } M = N \text{ then } P \text{ else } Q$	$\nu x.A$
$P \mid Q$	$\{^M/x\}$
$!P$	
$\nu n.P$	

The process 0 is the plain process that does nothing. The process $\text{in}(u, x).P$ waits to receive a message on the channel u , and then continues as P but with x replaced by the received message. The process $\text{out}(u, N).P$ outputs a term N on the channel u , and then continues as P . The process *if* $M = N$ *then* P *else* Q runs as P if the ground terms M and N are equal in the equational theory, and otherwise as Q . If there is no “else”, it means “else 0 ”. The process $P \mid Q$ runs P and Q in parallel. The process $!P$ executes P some finite number of times. The restriction νn is used to model the creation in a process of new random numbers (e.g., nonces or key material), or of new private channels. The process $\nu n.P$ is the process that invents a new name n and continues as P .

Extended processes add *active substitutions* (the process $\{^M/x\}$), restriction on names νn , and restriction on variables νx . Active substitutions are the notation that is used to denote a process that has output a term. Consider the process $\text{out}(c, N).P$, where c is a channel name, N is some term, and P is some continuation process. If $\text{out}(c, N).P$ is allowed to run in an environment, it will become the process $P \mid \{^N/x\}$, which means the process that can now run as P , and has output the term N . We do not retain the name of the channel name c , but we do give a handle name, here x , to the value that was output. The environment may now refer to the term N as x .

The handle x is important when the environment cannot itself describe the term that was output, except by referring to it as the term that was output (i.e., by the handle x). Consider the process $\nu k.\text{out}(c, \text{enc}(a, k)).P$ which creates a new key k and then outputs the name a encrypted with k . Here, a is a “free name” (modelling some well-known value) rather than a restricted name (like k) that was created by the process using the ν operator. The process $\nu k.\text{out}(c, \text{enc}(a, k)).P$ can output the term on the channel c , resulting in the process $\nu k.(P \mid \{\text{enc}(a, k)/x\})$. In this process, the environment has the term $\text{enc}(a, k)$, but it doesn’t have k since the process hasn’t output k . The environment can refer to the term $\text{enc}(a, k)$ as x .

The syntax of extended processes also allows restriction νx on variables x . The combination of νx and active substitutions generalise the familiar “let” operator from many functional programming languages. We define “let $x = M$ in P ” as an abbreviation of $\nu x.(\{^M/x\} \mid P)$.

A process can perform an input and then test the value of the input for equality (modulo \mathbf{E}) with some other term; for example, $\text{in}(u, x). \text{if } x = M$

then P . Suppose that after checking the input the process makes no further use of it (i.e., x does not occur in P). This idiom is quite common, so we abbreviate it as $\text{in}(u, =M).P$.

An *evaluation context* $C[_]$ is an extended process with a hole $_$ instead of an extended process; this is useful for describing part (e.g. the beginning) of a process, while leaving the hole to represent the other part that will be filled in later. Names and variables have scopes, which are delimited by restrictions νx and νn , and by inputs $\text{in}(u, x)$. We write $fv(A)$, $bv(A)$, $fn(A)$ and $bn(A)$ for the sets of free and bound variables and free and bound names of A , respectively. We also stipulate that, in an extended process, there is at most one substitution for each variable, and there is exactly one when the variable is restricted. We say that an extended process is *closed* if all its variables are either bound or defined by an active substitution.

2.2 Semantics

The operational semantics of processes in the applied pi calculus is defined by structural rules defining two relations: *structural equivalence*, noted \equiv , and *internal reduction*, noted \rightarrow .

Structural equivalence takes account of the fact that the syntax of processes necessarily makes distinctions that are not important. For example, $P \mid Q$ looks different from $Q \mid P$ but that difference is purely syntactic, and not important, so we say that $P \mid Q$ and $Q \mid P$ are structurally equivalent. Formally, structural equivalence is the smallest equivalence relation \equiv on extended processes that is closed under α -conversion on names and variables (that is, renaming a bound name or variable), application of evaluation contexts, and some other standard rules such as associativity and commutativity of the parallel operator and commutativity of the bindings. In addition the following three rules are related to active substitutions and equational theories.

$$\begin{aligned} \nu x. \{^M/x\} &\equiv 0 \\ \{^M/x\} \mid A &\equiv \{^M/x\} \mid (A\{^M/x\}) \\ \{^M/x\} &\equiv \{^N/x\} \quad \text{if } M =_{\mathbf{E}} N \end{aligned}$$

where, in the second equivalence, $A\{^M/x\}$ means A but with free occurrences of x replaced by M . Note the absence of the \mid . In $A\{^M/x\}$, the substitution is not an active substitution, but a normal “metasyntactic” substitution; it tells the reader to perform the substitution.

Example 3. Consider the following process P :

$$\nu s, k. (\text{out}(c_1, \text{enc}(s, k)) \mid \text{in}(c_1, y). \text{out}(c_2, \text{dec}(y, k)))$$

The first component publishes the message $\text{enc}(s, k)$ by sending it on c_1 . The second receives a message on c_1 , uses the secret key k to decrypt it, and forwards the resulting plaintext on c_2 . The process P is structurally equivalent to the following extended process A :

$$A = \nu s, k, x_1. (\text{out}(c_1, x_1) \mid \text{in}(c_1, y). \text{out}(c_2, \text{dec}(y, k)) \mid \{\text{enc}(s, k)/x_1\})$$

Internal reduction is the smallest relation on extended processes closed under structural equivalence and application of evaluation contexts such that

$$\begin{aligned}
 (\text{COMM}) \quad & \text{out}(a, x).P \mid \text{in}(a, x).Q \rightarrow P \mid Q \\
 (\text{THEN}) \quad & \text{if } M = M \text{ then } P \text{ else } Q \rightarrow P \\
 (\text{ELSE}) \quad & \text{if } M = N \text{ then } P \text{ else } Q \rightarrow Q
 \end{aligned}$$

for any ground terms M and N such that $M \neq_E N$.

This definition looks more restrictive than it is, thanks to structural equivalence. It is straightforward to prove that $\text{out}(a, M).P \mid \text{in}(a, x).Q \rightarrow P \mid Q\{M/x\}$ and if $M = N$ then $P \text{ else } Q \rightarrow P$ in the case that $M =_E N$.

The applied pi calculus has another kind of transition operation, called *labelled reduction*, denoted $\xrightarrow{\alpha}$, where α is a label. We don't define that formally here, but refer the reader to our full paper [9] or the applied pi calculus paper [2].

2.3 Observational Equivalence

Now we are able to define *observational equivalence*. This relation is important to understand how properties are defined in applied pi calculus. We write $A \Downarrow a$ when A can send a message on a , that is, when $A \rightarrow^* C[\text{out}(a, M).P]$ for some evaluation context $C[_]$ that does not bind a .

Definition 1. Observational equivalence (\approx) is the largest symmetric relation \mathcal{R} between closed extended processes with the same domain such that $A \mathcal{R} B$ implies:

1. if $A \Downarrow a$, then $B \Downarrow a$;
2. if $A \rightarrow^* A'$, then $B \rightarrow^* B'$ and $A' \mathcal{R} B'$ for some B' ;
3. $C[A] \mathcal{R} C[B]$ for all closing evaluation contexts $C[_]$.

Intuitively, two processes are observationally equivalent if they cannot be distinguished by any active attacker represented by any context.

Example 4. Let E be the theory defined by the axiom $\text{dec}(\text{enc}(x, y), y) = x$. Consider the processes $P_0 = \text{out}(c, \text{enc}(s_0, k))$ and $Q_0 = \text{out}(c, \text{enc}(s_1, k))$. We have that $\nu k.P_0 \approx \nu k.Q_0$; intuitively, the attacker cannot distinguish between the encryption of two known values s_0 and s_1 where the encryption is by a secret key. Technically, there is no context C that, given these processes, can distinguish them, e.g., by taking some observable action in the case of P_0 but not in the case of Q_0 . If the key k is available to the attacker, of course the situation changes. We have $P_0 \not\approx Q_0$, since the context

$$C[_] = \text{in}(c, x). \text{if } \text{dec}(x, k) = s_0 \text{ then } \text{out}(c, \text{"Found } s_0!\text{"}) \mid _$$

distinguishes P_0 and Q_0 .

Observational equivalence can be used to formalise many interesting security properties, in particular anonymity properties, such as those studied in this

paper (see Section 4). However, proofs of observational equivalences are difficult because of the universal quantification over all contexts. In [9], our definitions and proofs rely on *labelled bisimulation* which has been shown to coincide with observational equivalence [2]. Labelled bisimulation has the advantage of being more convenient to manipulate in proofs. Its definition relies on two further notions: *static equivalence* and *labelled reduction*. To avoid additional definitions and ease the presentation we stick to observational equivalence in this paper.

3 Formalising Voting Protocols

Before formalising security properties, we need to define what is an electronic voting protocol in applied pi calculus. Different voting protocols often have substantial differences. However, we believe that a large class of voting protocols can be represented by processes corresponding to the following structure.

Definition 2 (Voting process). *A voting process is a closed plain process*

$$VP \equiv \nu \tilde{n}. (V\sigma_1 \mid \dots \mid V\sigma_n \mid A_1 \mid \dots \mid A_m).$$

V is the template voter process, and the $V\sigma_i$ are the actual voter processes (the substitution σ_i provides the voter's identity). The A_j s are the election authorities that are required to be honest and the \tilde{n} are channel names. We also suppose that $v \in \text{dom}(\sigma_i)$ is a variable which refers to the value of the vote. We define an evaluation context S which is as VP , but has a hole instead of two of the $V\sigma_i$.

In order to prove a given property, we may require some of the authorities to be honest, while other authorities may be assumed to be corrupted by the attacker. The processes A_1, \dots, A_m represent the authorities which are required to be honest. The authorities under control of the attacker need not be modelled, since we consider any possible behaviour for the attacker (and therefore any possible behaviour for corrupt authorities). This arrangement implies that we consider only one attacker; to put in another way, we consider that all dishonest parties and attackers share information and trust each other, thus forming a single coalition. This arrangement does not allow us to consider attackers that do not share information with each other.

4 Formalising Privacy-Type Properties

In this section, we show how the anonymity properties, informally described in the introduction, can be formalised in our setting. Actually, it is rather classical to formalise anonymity properties as some kind of observational equivalence in a process algebra or calculus, going back to the work of Schneider and Sidiropoulos [15]. However, the definition of anonymity properties in the context of voting protocols is rather subtle.

4.1 Vote-Privacy

The privacy property aims to guarantee that the link between a given voter V and his vote v remains hidden. While generally most security properties should hold against an arbitrary number of dishonest participants, arbitrary coalitions do not make sense here. Consider for instance the case where all but one voter are dishonest: as the results of the vote are published at the end, the dishonest voter can collude and determine the vote of the honest voter. A classical device for modelling anonymity is to ask whether two processes, one in which V_A votes and one in which V_B votes, are equivalent. However, such an equivalence does not hold here as the voters' identities are revealed (and they need to be revealed at least to the administrator to verify eligibility). In a similar way, an equivalence of two processes where only the vote is changed does not hold, because the votes are published at the end of the protocol. To ensure privacy we need to hide the *link* between the voter and the vote and not the voter or the vote itself.

In order to give a reasonable definition of privacy, we need to suppose that at least two voters are honest. We denote the voters V_A and V_B and their votes a and b . We say that a voting protocol respects privacy whenever a process where V_A votes a and V_B votes b is observationally equivalent to a process where V_A votes b and V_B votes a . Formally, privacy is defined as follows.

Definition 3. *A voting protocol respects vote-privacy (or just privacy) if*

$$S[V_A\{^a/v\} \mid V_B\{^b/v\}] \approx S[V_A\{^b/v\} \mid V_B\{^a/v\}]$$

for all possible votes a and b .

The intuition is that if an intruder cannot detect if arbitrary honest voters V_A and V_B swap their votes, then in general he cannot know anything about how V_A (or V_B) voted. Note that this definition is robust even in situations where the result of the election is such that the votes of V_A and V_B are necessarily revealed. For example, if the vote is unanimous, or if all other voters reveal how they voted and thus allow the votes of V_A and V_B to be deduced.

As already noted, in some protocols the vote-privacy property may hold even if authorities are corrupt, while other protocols may require the authorities to be honest. When proving privacy, we choose which authorities we want to model as honest, by including them in Definition 2 of VP (and hence S).

4.2 Receipt-Freeness

Similarly to privacy, receipt-freeness may be formalised as an observational equivalence. However, we need to model the fact that V_A is willing to provide secret information, i.e., the receipt, to the coercer. We assume that the coercer is in fact the attacker who, as usual in the Dolev-Yao model, controls the public channels. To model V_A 's communication with the coercer, we consider that V_A executes a voting process which has been modified: inputs and freshly generated names of base type (i.e. not channel type) are forwarded to the coercer. We do

not forward restricted channel names, as these are used for modelling purposes, such as physically secure channels, e.g. the voting booth, or the existence of a PKI which securely distributes keys (the keys are forwarded but not the secret channel name on which the keys are received).

Definition 4. *Let P be a plain process and ch be a channel name. We define the process P^{ch} as follows:*

- $0^{ch} = 0$,
- $(P \mid Q)^{ch} = P^{ch} \mid Q^{ch}$,
- $(\nu n.P)^{ch} = \nu n.\text{out}(ch, n).P^{ch}$ when n is name of base type,
- $(\nu n.P)^{ch} = \nu n.P^{ch}$ otherwise,
- $(\text{in}(u, x).P)^{ch} = \text{in}(u, x).\text{out}(ch, x).P^{ch}$ when x is a variable of base type,
- $(\text{in}(u, x).P)^{ch} = \text{in}(u, x).P^{ch}$ otherwise,
- $(\text{out}(u, M).P)^{ch} = \text{out}(u, M).P^{ch}$,
- $(!P)^{ch} = !P^{ch}$,
- $(\text{if } M = N \text{ then } P \text{ else } Q)^{ch} = \text{if } M = N \text{ then } P^{ch} \text{ else } Q^{ch}$.

In the remainder, we assume that $ch \notin \text{fn}(P) \cup \text{bn}(P)$ before applying the transformation. Given an extended process A and a channel name ch , we need to define the extended process $A^{\backslash \text{out}(ch, \cdot)}$. Intuitively, such a process is as the process A , but hiding the outputs on the channel ch .

Definition 5. *Let A be an extended process.*

$$A^{\backslash \text{out}(ch, \cdot)} \hat{=} \nu ch.(A \mid \text{in}(ch, x)).$$

We are now ready to define receipt-freeness. Intuitively, a protocol is receipt-free if, for all voters V_A , the process in which V_A votes according to the intruder's wishes is indistinguishable from the one in which she votes something else. As in the case of privacy, we express this as an observational equivalence to a process in which V_A swaps her vote with V_B , in order to avoid the case in which the intruder can distinguish the situations merely by counting the votes at the end. Suppose the coercer's desired vote is c . Then we define receipt-freeness as follows.

Definition 6 (Receipt-freeness). *A voting protocol is receipt-free if there exists a closed plain process V' such that*

- $V'^{\backslash \text{out}(ch, \cdot)} \approx V_A\{a/v\}$,
- $S[V_A\{c/v\}^{chc} \mid V_B\{a/v\}] \approx S[V' \mid V_B\{c/v\}]$,

for all possible votes a and c .

As before, the context S in the second equivalence includes those authorities that are assumed to be honest. V' is a process in which voter V_A votes a but communicates with the coercer C in order to feign cooperation with him. Thus, the second equivalence says that the coercer cannot tell the difference between a situation in which V_A genuinely cooperates with him in order to cast the vote c and one in which she pretends to cooperate but actually casts the vote a ,

provided there is some counterbalancing voter that votes the other way around. The first equivalence of the definition says that if one ignores the outputs V' makes on the coercer channel chc , then V' looks like a voter process V_A voting a .

The first equivalence of the definition may be considered too strong. Informally, one might consider that the equivalence should be required only in a particular S context rather than requiring it in any context (with access to all the private channels of the protocol). This would result in a weaker definition, although one which is more difficult to work with. In fact, the variant definition would be only slightly weaker. It is hard to construct a natural example which distinguishes the two possibilities, and in particular it makes no difference to the case studies of later sections. Therefore, we prefer to stick to Definition 6.

Note that “receipt-freeness” does not preclude voting systems which give some kind of receipt to the voter that cannot be used for proving how she voted.

Intuition suggests an implication relation between receipt-freeness and vote-privacy, which indeed holds and is formally proved in 9:

If a protocol is receipt free (for a given set of honest authorities), then it also respects vote-privacy (for the same set).

5 Protocol Due to Fujioka, Okamoto and Ohta

In this section we study a protocol due to Fujioka, Okamoto and Ohta 12.

5.1 Description

The protocol involves voters, an administrator, verifying that only eligible voters can cast votes, and a collector, collecting and publishing the votes. In comparison with authentication protocols, the protocol also uses some unusual cryptographic primitives such as *secure bit-commitment* and *blind signatures*. Moreover, it relies on anonymous channels. We deliberately do not specify the way these channels are handled; most anonymiser mechanisms could be suitable depending on the precise context the protocol is used in. One can use MIX-nets introduced by Chaum 7 whose main idea is to permute and modify (by using decryption or re-encryption) some sequence of objects in order to hide the correspondence between elements of the original and the final sequences. Some other implementations may also be possible, e.g. onion routing 16.

A bit-commitment scheme allows an agent A to commit a value v to another agent B without revealing it immediately. Moreover, B is ensured that A cannot change her mind afterwards and that the value she later reveals will be the same as she thinks at the beginning. For this, A encrypts the value v in some way and sends the encryption to B . The agent B is not able to recover v until A sends him the key.

A blind signature scheme allows a requester to obtain a signature of a message m without revealing the message m to anyone, including the signer. Hence, the signer is requested to sign a message blindly without knowing what he signs. This mechanism is very useful in electronic voting protocol. It allows the voter

to obtain a signature of her vote by an authority who checks that she has right to vote without revealing it to the authority.

In a first phase, the voter gets a signature on a commitment to his vote from the administrator. To ensure privacy, blind signatures [8] are used, i.e. the administrator does not learn the commitment of the vote. (Throughout, we assume signatures with message recovery.)

- The voter V selects a vote v , computes the commitment $x = \xi(v, r)$ using a random key r , computes the message $e = \chi(x, b)$ using a blinding function χ and a random blinding factor b , and finally digitally signs e and sends her signature $\sigma_V(e)$ to the administrator A together with her identity.
- The administrator A verifies that V has the right to vote, has not voted yet and that the signature is valid; if all these tests hold, A digitally signs e and sends his signature $\sigma_A(e)$ to V ;
- V unblinds $\sigma_A(e)$ and obtains $y = \sigma_A(x)$, i.e. a signed commitment to V 's vote.

The second phase of the protocol is the actual voting phase.

- V sends y , A 's signature on the commitment to V 's vote, to the collector C using an anonymous channel;
- C checks correctness of the signature y and, if the test succeeds, enters (ℓ, x, y) into a list as an ℓ -th item.

The last phase of the voting protocol starts, once the collector decides that he received all votes, e.g. after a fixed deadline. In this phase the voters reveal the random key r which allows C to open the votes and publish them.

- C publishes the list (ℓ_i, x_i, y_i) of commitments he obtained;
- V verifies that her commitment is in the list and sends ℓ, r to C via an anonymous channel;
- C opens the ℓ -th ballot using the random r and publishes the vote v .

Note that we need to separate the voting phase into a commitment phase and an opening phase to avoid releasing partial results of the election and to ensure privacy. This is ensured by requiring synchronisation between the different agents involved in the election.

5.2 The Model in Applied pi

We only give the interesting parts of the modelling. A complete formalisation can be found in [9].

Cryptographic primitives as an equational theory. We model cryptography in a Dolev-Yao style as being perfect. The equations are given below.

$$\begin{aligned} \text{open}(\text{commit}(m, r), r) &= m \\ \text{checksign}(\text{sign}(m, \text{sk}), \text{pk}(\text{sk})) &= m \\ \text{unblind}(\text{blind}(m, r), r) &= m \\ \text{unblind}(\text{sign}(\text{blind}(m, r), \text{sk}), r) &= \text{sign}(m, \text{sk}) \end{aligned}$$

```

(* private channels *)
ν privCh.ν pkaCh1.ν pkaCh2.ν skaCh.ν skvaCh.ν skvbCh.
(* administrators *)
(processK | processA | processA | processC | processC |
(* voters *)
(let skvCh = skvaCh in let v = a in processV) |
(let skvCh = skvbCh in let v = b in processV) )

```

Process 1. Main process

In this model we can note that bit commitment (modelled by the functions `commit` and `open`) is identical to classical symmetric-key encryption. For simplicity, we identify host names and public keys. Our model of cryptographic primitives is an abstraction; for example, bit commitment gives us perfect binding and hiding. Digital signatures are modelled as being signatures with message recovery, i.e. the signature itself contains the signed message which can be extracted using the `checksign` function. To model blind signatures we add a pair of functions `blind` and `unblind`. These functions are again similar to perfect symmetric key encryption and bit commitment. However, we add a second equation which permits us to extract a signature out of a blind signature, when the blinding factor is known. Note that the equation modelling commitment cannot be applied on the term `open(commit(m, r1), r2)` when $r_1 \neq r_2$.

Process synchronisation. As mentioned, the protocol is divided into three phases, and it is important that every voter has completed the first phase before going onto the second one (and then has completed the second one before continuing to the third). We enforce this in our model by the keyword `synch`. When a process encounters `synch n`, it waits until all the other process that could encounter `synch n` arrive at that point too. Then all the processes are allowed to continue. If there are k processes that can encounter `synch n`, we can implement the synchronisation as follows. The command `synch n` is replaced by `out(n, 0); in(n, =1)` where `n` is a globally declared private channel. Moreover we assume an additional process `in(n, =0); ...; in(n, =0); out(n, 1); ...; out(n, 1)` that has k ins and k outs. This simple encoding is fine for our purpose since the value of k can be inferred by inspecting the code; it would not work if new processes were created, e.g. with “!”.

Main (Process 1). The main process sets up private channels and specifies how the processes are combined in parallel. Most of the private channels are for key distribution. We only model the protocol for two voters and launch two copies of the administrator and collector process, one for each voter.

Voter (Process 2). First, each voter obtains her secret key from the PKI as well as the public keys of the administrator. The remainder of the specification follows directly the informal description given in Section 5.1

```

process V =                                     (* parameters: skvCh, v *)
  (* her private key *)
  in (skvCh, skv).
  (* public keys of the administrator *)
  in (pkaCh1, pubka).
  v blinder. v r.
  let committedvote = commit(v, r) in
  let blindedcommittedvote = blind(committedvote, blinder) in
  out(ch1, (pk(skv), sign(blindedcommittedvote, skv))).
  in(ch2, m2).
  let result = checksign(m2, pubka) in
  if result = blindedcommittedvote then
  let signedcommittedvote = unblind(m2, blinder) in
  synch 1.
  out(ch3, (committedvote, signedcommittedvote)).
  synch 2.
  in(ch4, (l, =committedvote, =signedcommittedvote)).
  out(ch5, (l, r))

```

Process 2. Voter process

Our model also includes a dedicated process for generating and distributing keying material modelling a PKI (`processK`), a process for the administrator and another one for the collector (those processes are not given here, see [9]).

5.3 Analysis

Vote-privacy. According to our definition, to show that the protocol respects privacy, we need to show that

$$S[V_A\{^a/v\} \mid V_B\{^b/v\}] \approx S[V_A\{^b/v\} \mid V_B\{^a/v\}] \quad (1)$$

where $V_A = \text{processV}\{skvaCh/skvCh\}$, $V_B = \text{processV}\{skvbCh/skvCh\}$. We do not require that any of the authorities are honest, so they are not modelled in S , but rather left as part of the attacker context. However, we have to ensure that both voters use the same public key for the administrator. Therefore, we send this public key on a private channel (`pkaCh1`), although the public key and its counterpart are known by the attacker. Actually, we show that

$$\begin{aligned}
& \nu pkaCh1.(V_A\{^a/v\} \mid V_B\{^b/v\} \mid \text{processK}) \\
& \quad \approx \\
& \nu pkaCh1.(V_A\{^b/v\} \mid V_B\{^a/v\} \mid \text{processK})
\end{aligned} \quad (2)$$

The proof, detailed in [9], uses the (equivalent) definition of labelled bisimulation instead of observational equivalence. We were able to automate parts of the proof (the static equivalence relations) using the ProVerif tool [5]. The remaining of the proof (the bisimulation part) is established manually by considering

all cases. Although ProVerif provides observation equivalence checking, it was unable to perform the proof in this case: observation equivalence checking being undecidable, ProVerif aims at proving a finer relation which relies on easily matching up the execution paths of the two processes [6]. This relation happens to be too fine for proving equivalence [2]. Our proof relies on matching $V_A\{^a/v\}$ on the left-hand side with $V_A\{^b/v\}$ on the right-hand side during the first stage (before `synch 1`) and matching $V_A\{^a/v\}$ on the left with $V_B\{^a/v\}$ on the right in the following (after `synch 1`).

As mentioned above, the use of phases is crucial for privacy to be respected. When we omit the synchronisation after the registration phase with the administrator, privacy is violated. Indeed, consider the following scenario. Voter V_A contacts the administrator. As no synchronisation is considered, voter V_A can send his committed vote to the collector before voter V_B contacts the administrator. As voter V_B could not have submitted the committed vote, the attacker can link this commitment to the first voter's identity. This problem was found during a first attempt to prove the protocol where the phase instructions were omitted. The original paper divides the protocol into three phases but does not explain the crucial importance of the synchronisation after the first phase. Our analysis emphasises this need and we believe that it increases the understanding of some subtle details of the privacy property in this protocol.

Receipt-freeness. This scheme is not receipt-free and was in fact not designed with receipt-freeness in mind. Indeed, if the voter gives away the random numbers for blinding and commitment, i.e. b_A and r_A , the coercer can verify that the committed vote corresponds to the coercer's wish and by unblinding the first message, the coercer can trace which vote corresponds to this particular voter. Moreover, the voter cannot lie about these values as this will immediately be detected by the coercer. In our framework, this corresponds to the fact that there exists no V' such that:

- $V' \setminus \text{out}(chc, \cdot) \approx V_A\{^a/v\}$,
- $S[V_A\{^c/v\}^{chc} \mid V_B\{^a/v\}] \approx S[V' \mid V_B\{^c/v\}]$.

We have that $V_A\{^c/v\}^{chc}$ outputs the values r_A and b_A on the channel `chc`. This will generate entries in the frame. Hence, V' needs to generate similar entries in the frame. The coercer can now verify that the values r_A and b_A are used to encode the vote c in the message sent to the administrator. Thus V' is not able to commit to a value different from c , in order to satisfy the second equivalence. But then V' will not satisfy the first equivalence, since he will be unable to change his vote afterwards as the commitment to c has been signed by the administrator. Thus, the requirements on V' are not satisfiable.

Note that the failure of receipt-freeness is not due to the possible dishonesty of the administrator or collector; even if we include them as honest parties, the protocol still doesn't guarantee receipt-freeness. It follows that coercion-resistance doesn't hold either.

6 Protocol Due to Okamoto

In this section we study a protocol due to Okamoto [13] which was designed to be incoercible. However, Okamoto himself shows a flaw [14]. According to him, one of the reasons why the voting scheme he proposed had such a flaw is that no formal definition and proof of receipt-freeness and coercion-resistance have been given when the concept of receipt-freeness has been introduced by Benaloh and Tuinstra [4].

6.1 Description

The authorities managing the election are an administrator for registration, a collector for collecting the tokens and a timeliness member (denoted by T) for publishing the final tally. The main difference with the protocol due to Fujioka *et al.* is the use of a *trap-door bit commitment scheme* [10] in order to retrieve receipt-freeness. Such a commitment scheme allows the agent who has performed the commitment to open it in many ways. Hence, trap-door bit commitment does not bind the voter to the vote v . Now, to be sure that the voter does not change her mind at the end (during the opening stage) she has to say how she wants to open her commitment during the voting stage. This is done by sending the required information to T through an untappable anonymous channel, i.e. a physical apparatus by which only voter V can send a message to a party, and the message is perfectly secret to all other parties.

The first phase is similar to the one of the protocol due to Fujioka *et al.*. The only change is that ξ is a trap-door bit commitment scheme. The second phase of the protocol is the actual voting phase. Now, the voter has to say how she wants to open her commitment to the timeliness member T .

- V sends y , A 's signature on the trap-door commitment to V 's vote, to the collector C using an anonymous channel;
- C checks correctness of the signature y and, if the test succeeds, enters (x, y) into a list.
- V sends (v, r, x) to the timeliness member T through an untappable anonymous channel.

The last phase of the voting protocol starts, once the collector decides that he received all votes, e.g. after a fixed deadline.

- C publishes the list (x_i, y_i) of trap-door commitments he obtained;
- V verifies that her commitment is in the list;
- T publishes the list of the votes v_i in random order and also proves that he knows the permutation π and the r_i 's such that $x_{\pi(i)} = \xi(v_i, r_i)$ without revealing π or the r_i 's.

We have chosen to not entirely model this last phase. In particular, we do not model the zero-knowledge proof performed by the timeliness member T , as it is not relevant for illustrating our definitions of privacy, receipt-freeness and


```

(* private channels *)
ν privCh. ν pkaCh1. ν pkaCh2.
ν skaCh. ν skvaCh. ν skvbCh. ν chT.
(* administrators *)
(processK | processA | processA | processC | processC |
 processT | processT |
(* voters *)
(let skvCh=skvaCh in let v=a in processV) |
(let skvCh=skvbCh in let v=b in processV) )

```

Process 3. Main process

coercion-resistance. This proof of zero-knowledge is very useful to ensure that T outputs the correct vote chosen by the voter. This is important in order to ensure correctness, even in the case that T is dishonest. However, the proof of knowledge is unimportant for anonymity properties. In particular, if T is the coercer himself, then he can enforce the voter to vote as he wants as in the protocol due to Fujioka *et al.* Indeed, the timeliness member T can force the voter to give him the trap-door she has used to forge her commitment and then he can not only check if the voter has vote as he wanted, but he can also open her vote as he wants.

6.2 The Model in Applied pi

Cryptographic primitives as an equational theory. The equations modelling public keys and blind signatures are the same as in Section 5.2. To model trap-door bit commitment, we consider the two following equations:

$$\begin{aligned} \text{open}(\text{tdcommit}(m, r, \text{td}), r) &= m \\ \text{tdcommit}(m_1, r, \text{td}) &= \text{tdcommit}(m_2, f(m_1, r, \text{td}, m_2), \text{td}) \end{aligned}$$

Firstly, the term $\text{tdcommit}(m, r, \text{td})$ models the commitment of the message m under the key r by using the trap-door td . The second equation is used to model the fact that a commitment $\text{tdcommit}(m_1, r, \text{td})$ can be viewed as a commitment of any value m_2 . However, to open this commitment as m_2 one has to know the key $f(m_1, r, \text{td}, m_2)$. Note that this is possible only if one knows the key r used to forge the commitment $\text{tdcommit}(m_1, r, \text{td})$ and the trap-door td .

Main (Process 3). Again, the main process sets up private channels and specifies how the processes are combined in parallel. Most of the private channels are for key distribution. The channel chT is the untappable anonymous channel on which voters send to T how they want to open their commitment.

Voter (Process 4). This process is very similar to the one given in the previous section. We use the primitive tdcommit instead of commit and at the end, the voter sends, through the channel chT , how she wants to open her commitment.

```

process V =
  (* parameters: skvCh, v *)
  (* her private key *)
  in (skvCh, skv).
  (* public keys of the administrator *)
  in (pkCh1, pubka).
  v blinder. v r. v td.
  let committedvote = tdcommit(v, r, td) in
  let blindedcommittedvote = blind(committedvote, blinder) in
  out(ch1, (pk(skv), sign(blindedcommittedvote, skv))).
  in(ch2, m2).
  let result = checksign(m2, pubka) in
  if result = blindedcommittedvote then
  let signedcommittedvote = unblind(m2, blinder) in
  synch 1.
  out(ch3, (committedvote, signedcommittedvote)).
  out(chT, (v, r, committedvote))

```

Process 4. Voter process

```

process T =
  synch 1.
  (* reception du commitment *)
  in (chT, (vt, rt, xt)).
  synch 2.
  if open(xt, rt) = vt then
  out(board, vt)

```

Process 5. Timeliness process

Timeliness Member (Process 5). The timeliness member receives, through chT , messages of the form (vt, rt, xt) where vt is the value of the vote, xt the trap-door bit commitment and rt the key he has to use to open the commitment. In a second phase, he checks that he can obtain vt by opening the commitment xt with rt . Then, he publishes the vote vt on the board. This is modelled by sending vt on a public channel.

We have also a dedicated process for generating and distributing keying material modelling a PKI, an administrator process and a collector. Those processes are not given here.

6.3 Analysis

Unfortunately, the equational theory which is required to model this protocol is beyond the scope of ProVerif and we cannot rely on automated verification even for the static equivalence parts. Thus, our analysis is entirely manual. We only discuss receipt-freeness (since this implies vote privacy).

```

processV =
  (* her private key *)
  in (skvCh, skv). out (chc, skv).
  (* public keys of the administrator *)
  in (pkaCh1, pubka). out (chc, pubka).
  ν blinder. ν r. ν td.
  out (chc, blinder). out (chc, f(a, r, td, c)). out (chc, td).
  let committedvote = tdcommit(a, r, td) in
  let blindedcommittedvote = blind(committedvote, blinder) in
  out (ch1, (pk(skv), sign(blindedcommittedvote, skv))).
  out (chc, (pk(skv), sign(blindedcommittedvote, skv))).
  in (ch2, m2).
  let result = checksign(m2, pubka) in
  if result = blindedcommittedvote then
  let signedcommittedvote = unblind(m2, blinder) in
  synch 1.
  out (ch3, (committedvote, signedcommittedvote)).
  out (chc, (committedvote, signedcommittedvote)).
  out (chT, (a, r, committedvote)).
  out (chc, (c, f(a, r, td, c), committedvote))

```

Process 6. V' - Receipt-freeness

Receipt-freeness. To establish receipt-freeness one needs to construct a process V' which successfully fakes all secrets to a coercer. The idea is for V' to vote a , but when outputting secrets to the coercer, V' lies and gives him fake secrets to pretend to cast the vote c . The crucial part is that, using trap-door commitment and thanks to the fact that the key used to open the commitment is sent through an untappable anonymous channel, the value given by the voter to the timeliness member T can be different from the one she provides to the coercer. Hence, the voter who forged the commitment, provides to the coercer the one allowing the coercer to retrieve the vote c , whereas she sends to T the one allowing her to cast the vote a .

We describe such a process V' in Process 6. To prove receipt-freeness, we need to show that

- $V' \setminus \text{out}(chc, \cdot) \approx V_A\{a/v\}$, and
- $S[V_A\{c/v\}^{chc} \mid V_B\{a/v\}] \approx S[V' \mid V_B\{c/v\}]$.

The context S we consider is $\nu pkaCh1.\nu chT.(- \mid \text{processK} \mid \text{processT} \mid \text{processT})$. The first equivalence may be seen informally by considering V' without the instructions “out(chc, ...)”, and comparing it visually with $V_A\{a/v\}$. The two processes are the same. To see the second labelled bisimulation, we have to consider all the executions of each side. As before, the details may be found in [9].

7 Conclusion

We have defined a framework for modelling cryptographic voting protocols in the applied pi calculus, and shown how to express in it the properties of vote-privacy, receipt-freeness and coercion-resistance. Within the framework, we can stipulate which parties are assumed to be trustworthy in order to obtain the desired property. We investigated two protocols from the literature. Our results are summarised in Figure 1.

<i>Property</i>	<i>Fujioka et al.</i>	<i>Okamoto et al.</i>
Vote-privacy trusted authorities	✓ none	✓ timeliness mbr.
Receipt-freeness trusted authorities	× n/a	✓ timeliness mbr.

Fig. 1. Summary of protocols and properties

We have stated the intuitive relationships between the two properties: for a fixed set of trusted authorities, receipt-freeness implies vote-privacy. This is proved in our full version [9].

Some of our reasoning about bisimulation in applied pi has been informal. In the future, we hope to develop better techniques for formalising and automating this reasoning.

Acknowledgments. Thanks to Michael Clarkson and Olivier Pereira for interesting questions and discussions. Thanks also to the editors of this volume for detailed comments about the presentation, which helped us improve the readability.

References

1. Abadi, M., Blanchet, B., Fournet, C.: Just fast keying in the pi calculus. In: Schmidt, D. (ed.) ESOP 2004. LNCS, vol. 2986, pp. 340–354. Springer, Heidelberg (2004)
2. Abadi, M., Fournet, C.: Mobile values, new names, and secure communication. In: Proc. 28th ACM Symposium on Principles of Programming Languages (POPL 2001), London, UK, pp. 104–115. ACM, New York (2001)
3. Abadi, M., Gordon, A.D.: A calculus for cryptographic protocols: The spi calculus. In: Proc. 4th ACM Conference on Computer and Communications Security (CCS 1997), pp. 36–47. ACM Press, New York (1997)
4. Benaloh, J., Tuinstra, D.: Receipt-free secret-ballot elections (extended abstract). In: Proc. 26th Symposium on Theory of Computing (STOC 1994), pp. 544–553. ACM Press, New York (1994)
5. Blanchet, B.: An efficient cryptographic protocol verifier based on prolog rules. In: Proc. 14th IEEE Computer Security Foundations Workshop (CSFW 2001), pp. 82–96. IEEE Comp. Soc. Press, Los Alamitos (2001)

6. Blanchet, B., Abadi, M., Fournet, C.: Automated Verification of Selected Equivalences for Security Protocols. In: Proc. 20th IEEE Symposium on Logic in Computer Science (LICS 2005), pp. 331–340. IEEE Comp. Soc. Press, Los Alamitos (2005)
7. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM* 24(2), 84–88 (1981)
8. Chaum, D.: Blind signatures for untraceable payments. In: *Advances in Cryptology – CRYPTO 1982*, pp. 199–203. Plenum Press, New York (1983)
9. Delaune, S., Kremer, S., Ryan, M.D.: Verifying privacy-type properties of electronic voting protocols. Research report, Laboratoire Spécification et Vérification, ENS Cachan, France (January 2008)
10. Fischlin, M.: Trapdoor Commitment Schemes and Their Applications. PhD thesis, Fachbereich Mathematik Johann Wolfgang Goethe-Universität Frankfurt am Main (2001)
11. Fournet, C., Abadi, M.: Hiding names: Private authentication in the applied pi calculus. In: Okada, M., Pierce, B.C., Scedrov, A., Tokuda, H., Yonezawa, A. (eds.) *ISSS 2002*. LNCS, vol. 2609, pp. 317–338. Springer, Heidelberg (2003)
12. Fujioka, A., Okamoto, T., Ohta, K.: A practical secret voting scheme for large scale elections. In: Zheng, Y., Seberry, J. (eds.) *AUSCRYPT 1992*. LNCS, vol. 718, pp. 244–251. Springer, Heidelberg (1993)
13. Okamoto, T.: An electronic voting scheme. In: Proc. IFIP World Conference on IT Tools, pp. 21–30 (1996)
14. Okamoto, T.: Receipt-free electronic voting schemes for large scale elections. In: Christianson, B., Lomas, M. (eds.) *Security Protocols 1997*. LNCS, vol. 1361, pp. 25–35. Springer, Heidelberg (1998)
15. Schneider, S., Sidiropoulos, A.: CSP and anonymity. In: Martella, G., Kurth, H., Montolivo, E., Bertino, E. (eds.) *ESORICS 1996*. LNCS, vol. 1146, pp. 198–218. Springer, Heidelberg (1996)
16. Syverson, P.F., Goldschlag, D.M., Reed, M.G.: Anonymous connections and onion routing. In: Proc. 18th IEEE Symposium on Security and Privacy (SSP 1997), pp. 44–54. IEEE Comp. Soc. Press, Los Alamitos (1997)

Improving Remote Voting Security with CodeVoting

Rui Joaquim, Carlos Ribeiro, and Paulo Ferreira

ISEL - Technical University of Lisbon - INESC-ID
rjoaquim@cc.isel.ipl.pt,
carlos.ribeiro@tagus.ist.utl.pt,
paulo.ferreira@inesc-id.pt

Abstract. One of the major problems that prevents the spread of elections with the possibility of remote voting over electronic networks, also called Internet Voting, is the use of unreliable client platforms, such as the voter's computer and the Internet infrastructure connecting it to the election server. A computer connected to the Internet is exposed to viruses, worms, Trojans, spyware, malware and other threats that can compromise the election's integrity. For instance, it is possible to write a virus that changes the voter's vote to a predetermined vote on election's day. Another possible attack is the creation of a fake election web site where the voter uses a malicious vote program on the web site that manipulates the voter's vote (phishing/pharming attack). Such attacks may not disturb the election protocol, therefore can remain undetected in the eyes of the election auditors.

We propose the use of CodeVoting to overcome insecurity of the client platform. CodeVoting consists in creating a secure communication channel to communicate the voter's vote between the voter and a trusted component attached to the voter's computer. Consequently, no one controlling the voter's computer can change the his/her's vote. The trusted component can then process the vote according to a cryptographic voting protocol to enable cryptographic verification at the server's side.

Keywords: Remote voting, Internet voting, vote manipulation, uncontrolled voting platform, insecure voting platform.

1 Introduction

Remote electronic voting over electronic networks, also called Internet voting, is very appealing because it offers the possibility of voting from anywhere with an Internet connection, on election day, avoiding the need to vote in advance. Therefore, remote Internet voting offers a convenient way to vote for users away from home on vacations, due to work, or for any other expected or unexpected reason. However, surprisingly or not, there are not many Internet voting systems in use today. One of the main reasons for this situation is that a computer connected to the Internet does not offer a secure voting environment.

A computer connected to the Internet is exposed to several threats, such as viruses, worms and Trojans, among others. These threats take advantage of vulnerabilities in software to perform malicious actions, including taking control of the computer. The number of new vulnerabilities reported is too high to be ignored, 8064 in 2006 and 5568 in the first three quarters of 2007 [1,2]. It is easy to imagine a virus exploring one of those vulnerabilities to steal or change the voter's vote.

It is also possible, in some situations, to steal the voter's vote without attacking the voter's computer. An attack to the Internet support infrastructure could send the voter to a fake election web site. Once in the fake election web site the attacker could use the voter's authentication information and steal the voter's vote. Such type of attack is called pharming [3]. An example of an attack technique against home routers can be found in [4]. Reports on recent pharming attacks against banks in the US, Europe and Asia-Pacific can be found in [5,6].

Another important issue concerning remote voting in general is that there are no guarantees that the voter will vote alone. Therefore, the voter may be subject to coercion. A special case of coercion within family members that live in the same house, named family voting, has been observed in several countries [7] and is a problem to consider whenever remote voting is allowed, e.g. postal voting, Internet voting. A problem that is somehow similar to coercion is vote selling. Since the voters cast votes in an uncontrolled environment they can give their ballot to anyone, or vote in the presence of anyone, therefore they may be tempted to sell their vote. A possible protection against vote selling and coercion is to allow the voter to update their vote, i.e. cast several votes [8].

The main difference between traditional remote voting, e.g. postal voting, and Internet voting is that Internet voting attacks are able to target a large number of voters with a fraction of the budget. Our opinion is that an attack to steal/change the voter's vote by attacking the voter's computer or the Internet infrastructure poses a potentially higher risk to the election's integrity than an online vote buying or coercion attack. We base our opinion on the following four reasons:

- First, large scale vote buying/coercion, involving possibly thousands of voters, is quite unlikely to pass undetected. Additionally, vote buying/coercion can be discouraged by allowing vote updates.
- Second, with all the security flaws in operating systems and applications, it is easy to write a virus that would be active on election day to change the voter's vote.
- Third, we believe that writing a virus and disseminating it would be cheaper and more difficult to trace back to the authors than a vote buying/coercion attempt of a thousand voters, therefore, more appealing to an attacker.
- Fourth, punishing the attackers would be very difficult, if not impossible, because the attack could be carried out from anywhere in the world.

Therefore, we can say that the insecure voting platform [9,10,11,12,13], offered by a computer connected to the Internet, and the possibility of an unpunished attack are the main issues that prevent the spread of remote Internet voting.

We propose the use of CodeVoting, a technique to communicate the voter's vote between the voter and a trusted component attached to the voter's computer. Consequently, no one controlling the voter's computer can change the vote. CodeVoting offers a manually verifiable proof of correct vote fulfillment and submission, based on the assumption about the existence of a trusted and secure component attached to the voter's computer. In this paper we present the CodeVoting technique [14,15] and an enhancement to support large candidate lists and multiple elections.

1.1 Why Cryptographic Voting Protocols Are Not Enough

In the last 30 years many cryptographic voting protocols were proposed to secure electronic voting. Several cryptographic techniques were employed such as blind signatures [16,17,18,19], mix-nets [20,21,22,23] and homomorphic ciphers [24,25,26,27]. However, the main concern of these cryptographic voting protocols is the protection against vote manipulation at server side, assuming a trusted client to establish a secure communication channel to the election server and to perform the cryptographic steps and verification of the voting protocol.

It is clear that these cryptographic voting protocols do not work well without the assumption of a trusted voting client. The CodeVoting's goal is to provide the means to secure the use of a cryptographic voting protocol from a generic computer connected to the Internet.

1.2 CodeVoting Overview

CodeVoting does not replace traditional voting protocols. It works by creating a secure channel between the user and a trusted component that runs one of the traditional voting protocols with the voting server (Fig. 1). Therefore, CodeVoting can be considered a kind of user interface for the voting system.

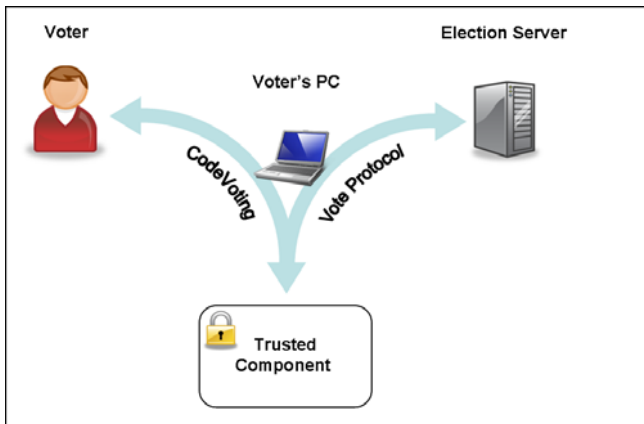


Fig. 1. CodeVoting overview

We propose the use of a tamper resistant device, such as a smart card, to be the trusted component of the voting system at the client's side. Since a smart card provides a much more secure execution platform than an off-the-shelf PC, using CodeVoting to create a secure communication channel to the smart card, through the voter's PC, will prevent automatic vote manipulation by malicious software installed in the voter's PC. Since we propose the use of a cheap tamper resistant device, such as a smart card, that does not have network nor I/O capabilities, the voter's PC network and I/O capabilities will still be used to interact with the voter and the server's side of the voting system.

The use of secure devices, e.g. smart cards, in electronic voting is not a new concept [28,29]. Secure devices are typically used as a way to provide secure voter's authentication and for the generation and secure storage of secret values for a voting protocol. However, it is always necessary to use the voter's computer to show the ballot and collect the answer, and this is usually assumed to be performed by a trusted vote client application. Our goal is to show how one can build a simple, secure and private communication channel between the voters and their trusted devices, without the need to trust in the voters' computers. Secure communication channels are easy to achieve between machines, e.g. by sharing a secret key. However, making a secure and private communication channel between a machine and a human is not so straightforward. The challenge is to keep the complexity of the communication channel as small as possible in order for the human to be able to deal with it.

1.3 Vote Manipulation Attacks

A vote manipulation attack can be a modification of the voter's vote. It can be performed in two ways: i) changing a vote to a predetermined candidate, or ii) changing a vote to a random candidate. While the first attack is more powerful, the second may be easier to prepare in advance. In other words, to change a vote to a predetermined candidate one must have the knowledge of which candidate one wants to change the vote to, while to change the vote to a random candidate there is no need to know the candidates in advance.

If one wants to increase the number of votes for a candidate A it is preferable to perform an attack that will directly change the votes to votes for candidate A. On the other hand, if one wants to decrease the number of votes for a candidate B, it suffices to perform a random vote modification attack in an area known to be more favorable to candidate B.

The other kind of vote manipulation attack is to fake a successful vote delivery. In many voting systems this can be done by just presenting the message "Your vote was successfully delivered. Thank you for voting.". This attack allows an attacker to reduce the votes on a candidate just by targeting an area with great affinity for that candidate.

In the next section we describe the currently proposed approaches to minimize the weaknesses at the client's side. In section 3 we present the CodeVoting technique, a solution that prevents vote manipulation at the client's side by using a mix of "special security PC hardware" and "code sheet" approaches. We evaluate

the CodeVoting proposal's resistance to vote manipulation attacks in section [4](#). In section [5](#) we present the Matrix CodeVoting, a CodeVoting enhancement to support large candidate lists and multiple elections. Finally, in section [6](#), we present the conclusions and future work.

2 Related Work

Cryptographic voting protocols can prevent vote manipulation at server side. However, that is only relevant if the votes cannot be easily manipulated at the uncontrolled client side of a remote voting system. Here we present an overview of the approaches proposed to mitigate the problem of the unreliable vote client platform [9,30](#).

2.1 Clean Operating System and Voting Application

This approach assumes the existence of a CD-ROM with a “clean” and certified operating system and voting application. The voter should boot her computer from the CD-ROM to have access to the vote application. One of the major problems with this approach is how to design such CD-ROM so that it would allow the voters to boot from any computer in use. Another problem is how to provide Internet access. Voters have different types of Internet connections at home, such as modem, ADSL, cable. Would the voters have to manually configure their connection parameters?

The immediate consequence of such variety of configurations is that a large amount of software, besides the operating system and voting application, has to be on the CD-ROM, and consequently also, has to be verified. Therefore, some questions remain: can we claim that a CD-ROM is clean? And to which extent? Is it clean enough to provide a secure voting platform?

With all these questions/problems pending we do not believe that this approach could be successfully used to enable secure voting from any computer with an Internet connection.

2.2 Special Secure Hardware for a PC

This approach assumes a dedicated software-closed security device, with secure I/O, attached to the voter's computer, e.g. through a USB port. Its purpose is to display the ballot to the user, accept the voter's choices as input, and perform cryptographic operations. In effect, the voting protocol is executed by the secure device. Since the device is software-closed, meaning its software cannot be changed, it is not subject to infection with a malicious code. The main disadvantage of this approach is the cost of such dedicated hardware. Moreover, the manufacturer and the distribution process of the devices must be fully trusted.

Zúquete et al. [31](#) implemented a system based on this concept. They use a secure smart card terminal with I/O capabilities to display the ballot and read the voter's answer. In addition, they use a smart card to provide public key authentication. The main disadvantage of the system, besides the cost, is the reduced display capacity of the terminal, which is only 4 lines of 20 characters.

2.3 Closed Secure Devices

The use of closed secure devices was one of the proposals made by the California Internet Voting Task Force in 2000 [9]. The proposal was the use of special software-closed and Internet-capable devices, such as network computers (NCs) or hand-held, wireless descendants of those days (early 2000s) cell phones and electronic organizers.

However, modern cell phones and electronic organizers that have Internet access usually allow the user to install arbitrary applications too. This facility makes the systems open to malicious applications that take advantage of the vulnerabilities of the operating system of the device.

Moreover, the use of closed secure devices just to access a web site on the Internet, through which the voter votes, is vulnerable to attacks to the Internet infrastructure, such as pharming attacks.

2.4 Secure PC Operating Systems

This approach suggests the use of secure PC operating systems that may be composed of digitally-signed modules, allowing secure applications to exclude, as untrusted, modules of dubious origins (i.e. potentially malicious programs).

Trusted computing is the name given to the technology that is being developed today to offer such secure platform support [32]. Trusted computing is a technology that allows the remote attestation of machines and programs running on them. With remote attestation it is possible to certify that a voter is using a correct voting program. Trusted computing also provides ways to secure I/O operations between the program and the physical I/O devices, therefore creating a secure environment for an application to run. The attestation process is based on measures performed on the software by a hardware module called trusted platform module (TPM).

The client of a remote voting application needs to interact with the voter (I/O device drivers), needs to establish a connection with a voting server (network protocol stack + network adapter driver) and, last but not least, it needs an environment to run on, i.e. a working operating system. The attestation of the core of the operating system, the device drivers and the voting application can be cumbersome. Moreover, there are also problems concerning the maturity of the currently deployed technology [33] and concerning the revocation of cracked machines [34]. We believe that, for the time being, the application of trusted computing to remote voting as the only guarantee of the correct application behavior is not a valid alternative. Nevertheless, there are proposals to use trusted computing technology to solve the uncontrolled platform problem in remote voting [35].

2.5 Code Sheets

This approach consists in secretly sending, e.g. by mail, code sheets to voters that map their choices to entry codes on their ballot. While voting, the voter uses

the code sheet to know what to type in order to vote for a particular candidate. In effect, the voter does the vote encryption and, since no malicious software on the PC has access to the code sheet, it is not able to change a voter's intentions.

The first code sheet implementation we are aware of was proposed in 2001 by Chaum [36], the SureVote system. SureVote allows the voter to cast a vote using secret vote and reply codes. SureVote generates secret vote and reply codes for each candidate and for each voter. The codes are delivered to the voters prior to the election day. On election day the voter sends the vote code of her/his favorite candidate through the voting channel, e.g. Internet. At server side, the reply code is computed by a set of trustees and sent to the voter that confirms it - in this way it is verified that there was no vote modification. After the election day the trustees compute the real votes from the vote codes and publish the results. However, if there is at least one corrupted trustee, SureVote does not guarantee that in the counting phase the vote code is translated to the right candidate.

A code sheet system was used in the UK on some pilots of Internet, SMS and telephone voting [37,38]. A similar system was also proposed by Helbach and Schwenk [39] in which they suggest the use of a three-way-handshake voting protocol. They use a third code to confirm the vote.

The main drawback of this approach is the difficulty to guarantee that the codes are secretly generated and anonymously delivered to the voters. Another drawback is that there is no guarantee that the code vote is translated to the right candidate at server side, i.e. the reply code only confirms that the vote has reached an entity that knows the right reply code.

2.6 Test Ballots

This approach requires the use of special test ballots to be sent from clients and checked by software at the election authorities' office. The number, location, timing, and contents of the test ballots should be known by the county, but they should be otherwise indistinguishable from real ballots, so that any malicious code that destroys or changes real ballots will affect the test ballots as well. The analysis of the test ballots will enable any malicious code attacks to be detected, the locations of infected machines to be determined, the approximate time of the attack to be estimated, and the total number of votes affected to be bounded. Note that this technique does not prevent malicious code attacks; it only detects them after their occurrence. Hence it must be combined with one of the previously presented techniques.

Of course, this technique only works if the attack is to be performed after the vote is produced by the vote client software. The attack will not be noticed if it just modifies the voter's option before passing it to the method that processes the vote and delivers it to the vote server.

Still, this approach can be used as a kind of intrusion detection system that can detect any systematic cause of lost ballots, not just malicious code attacks, and provide a quantitative measure of the size of any problem it detects.

2.7 Obscurity/Complexity

This approach, while not sufficient to guarantee the security of the system, raises the cost for potential attackers. Digital ballot formats and voting software may be kept secret prior to the election and possibly randomly changed during the election, or made complex in other ways. In order to successfully carry out an attack and escape detection, malicious software authors must have a great deal of information about the internal format of the ballot and voting software. If these details are not available in advance, and/or if that information is complex, the potential authors of attack software may not have enough time to develop and distribute it during the election window.

On the other hand, it is difficult to establish a lower bound to the time needed to write malicious software. Additionally, the system is still vulnerable to pharming attacks that collect voters' authentication data and use them later in the real voting client software.

2.8 External Channel Verification

This approach consists in having a secondary communication channel to the election server that would allow the confirmation of the correct vote delivery. Kutylowski and Zagórski [40] proposed a voting protocol where a voter uses a secondary channel first to “decrypt” the ballot and choose the candidate, and then to verify with a probability of $\frac{1}{2}$ that the vote was correctly submitted to the election server. The main disadvantage of this protocol is the complexity of the verification tasks given to the voter that must deal directly with the encrypted ballots.

Skagestein et al. [41] proposed the verification of the clear casted vote. In their approach, a voter who wants to verify her/his vote can just use another PC and ask to see the casted vote. This second PC asks the voting server for the voter's vote, opens it with the secret encryption key used to encrypt the vote that should be stored in a secure medium at the time of vote casting, and displays the vote to the voter. To minimize the danger of vote selling and coercion, the authors proposed that the cast of several votes should be allowed; therefore, the vote buyer or coercer would not know if the verified vote was the final one. The main disadvantage of this protocol is that it does not prevent vote manipulation at server side. Additionally, anyone with access to the encrypted ballots, considered to be part of the final tally, can use them as a proof of vote since they can be decrypted using the secret encryption keys kept by the voters.

The main disadvantage of this approach is that it requires the voter to have access to two independent communication channels. Additionally, a verification step sometime after the vote casting procedure is not convenient for voters.

3 CodeVoting

We propose CodeVoting, a solution/system to prevent vote manipulation at client side while allowing the use of cryptographic voting protocols to protect

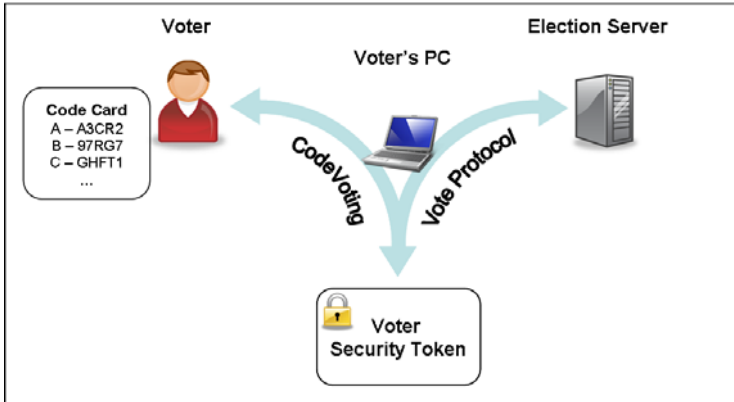


Fig. 2. CodeVoting components' overview

the election's integrity at server side. CodeVoting is a mix of the “special security PC hardware” and “code sheet” approaches, cf. sections 2.2 and 2.5. Figure 2 presents an overview of the CodeVoting components.

CodeVoting proposes the use of a tamper resistant device without any human I/O capabilities, the Voter Security Token (VST), to perform a cryptographic voting protocol at client side and securely authenticate the voter, e.g. by means of digital signatures. The voter communicates her/his choice to the VST using a code sheet approach based on codes printed on a paper card, the CodeCard. The CodeCard is generated directly by the VST, which prevents vote manipulation by a malicious computer. On the other hand, as explained later, the VST also provides a proof of the correct conclusion of the voting protocol.

Briefly, our solution consists in the following steps: i) the voter expresses her/his vote as a secret code, ii) the secret code is translated into the corresponding candidate identifier (ID) (clear vote), iii) the clear vote is used in a cryptographic voting protocol, and iv) once a cryptographic proof of the correct vote delivery is received, a successful vote delivery code is released to the voter.

3.1 Voter Security Token

The VST is the component in charge of the vote code translation to the clear vote as printed in the CodeCard (Sec. 3.2). After the vote code translation it is possible to use it in any voting protocol. The voting protocol runs inside the VST to prevent any vote manipulation at client side. It is possible to use a cryptographic voting protocol to prevent vote manipulation at server side. Usually, cryptographic voting protocols require the use of digital signatures to authenticate the voters. We suggest the use of the VST to enable such authentication mechanism. The VST should be protected by a PIN to prevent unauthorized access.

We propose to distribute the VSTs to the voters in a preliminary registration phase. This procedure is only required once, i.e. the VST will be reused in subsequent elections. To provide a secure voter's authentication mechanism, by

means of a digital signature, a public key infrastructure (PKI) should be in place before the registration process. The PKI can be set up just for election purposes or can be more widely used in a national e-Government project. This last approach can be useful to prevent, at some level, vote buying and coercion, because if the voter gives her/his VST to a vote buyer/coercer it is not just a vote that the s/he gives away, it is also all the e-Government rights of the voter.

3.2 CodeCard

The CodeCard is nothing more than a paper card that associates each candidate ID to a vote code printed on it. There should be one CodeCard per VST so that every voter votes with different codes. The voter should be the only one with access to the codes printed on her/his CodeCard. Consequently, we have a problem to solve: how to create the CodeCard, associate it with the VST and give it to the voter without leaking the codes.

We propose to generate each voter's CodeCard within the VST. This is a viable option because the CodeCard becomes automatically associated with the corresponding VST and no other entity besides the VST has access to the codes on the CodeCard. However, we still have the problem of how to secretly print the CodeCard, i.e. how to give it to the voter without leaking the codes. We believe the best idea is to have a certified CodeCard printing machine, the CodeCard generator interface (CCGI), available at the local authorities' offices. Since the codes are generated inside the VST, the CCGI would be very simple. It would consist of only an interface to the VST, e.g. a smart card reader, in the case of using smart cards, a keypad (for inserting a PIN to unlock the VST) and a small printer. We believe that such simple hardware could be easily certified and sealed to ensure the secrecy of the codes printed. With a certified CCGI in all local authorities' offices a voter can go to any local authority's office and generate a new CodeCard for her/his VST. For privacy reasons the CCGI should be inside a private booth, similar to the ones used for traditional paper-based voting.

3.3 CodeVoting Details

CodeVoting can be seen as a rearrangement of the ideas presented by Chaum [36]. However, the idea of CodeVoting is to only use the codes as a user interface and not as the entire voting protocol. The secret codes are the base for the secure communication channel between the voter and her/his VST. The voter uses secret codes to choose her/his favorite candidate. Each VST has a set of secret codes associated with it that are printed on a CodeCard.

For the voter, the voting process is quite simple. The voter just uses a CodeCard to translate the candidate ID into a vote code. For instance, a voter, with the ballot and CodeCard of Fig. 3, who wishes to vote for a candidate D only has to enter WL764 as the vote code.

Every voter will have a different CodeCard. Therefore, different vote codes exist for the same candidate. Each CodeCard is associated with a VST, which is responsible for the translation of the vote code to the candidate ID. Only the

<p>Election for the most important figure in security.</p> <p>A - Alice B - Bob C - Eavesdropper D - Attacker</p> <p>Enter your vote code:</p>	<p>CodeCard</p> <table border="1" style="width: 100%; border-collapse: collapse; margin-bottom: 10px;"> <thead> <tr> <th style="text-align: left;">Candidate</th> <th style="text-align: left;">Vote Code</th> </tr> </thead> <tbody> <tr> <td>Blank Vote</td> <td>SIT5Y</td> </tr> <tr> <td>A</td> <td>A3CR2</td> </tr> <tr> <td>B</td> <td>97RG7</td> </tr> <tr> <td>C</td> <td>GHFT1</td> </tr> <tr> <td>D</td> <td>WL764</td> </tr> <tr> <td>...</td> <td></td> </tr> </tbody> </table> <p>Confirmed vote delivery code: 6HKG2</p>	Candidate	Vote Code	Blank Vote	SIT5Y	A	A3CR2	B	97RG7	C	GHFT1	D	WL764	...	
Candidate	Vote Code														
Blank Vote	SIT5Y														
A	A3CR2														
B	97RG7														
C	GHFT1														
D	WL764														
...															

Fig. 3. Example of a ballot (on the left) and a CodeCard (on the right)

voter and the VST should know the codes written on the CodeCard. Therefore, CodeVoting is able to prevent a malicious voting application from changing the voter's vote. Note that there should also be a specific code for a blank or spoiled vote to prevent the malicious voting application from easily casting such a vote.

After translating the vote code to the candidate ID, any voting protocol can be used to cast the vote, e.g. a cryptographic voting protocol that protects the election's integrity at server side. When the VST receives a confirmation of a successful vote delivery from the election server, it releases the confirmed vote delivery code, assuring the voter that her vote was successfully delivered.

Based on the description of CodeVoting the reader can easily understand that CodeVoting is a type of user interface plugin to a voting system that protects the voter's choice from manipulation.

Note that we do not use different reply codes for each candidate. The reason for this is simple, as explained in section 2.5, the code sheet approach offers no guarantees that the vote code is translated to the right candidate at server side, i.e. the reply code only confirms that the vote has reached an entity that knows the right reply code. Therefore, the only advantage of using a different reply code for each candidate is that it makes it more difficult for an attacker to change the vote to a random candidate without being detected by the voter, i.e. the attacker needs to get the correct vote and reply codes.

However, to avoid vote stealing, the length of the vote codes must be defined to prevent an attacker from guessing a valid vote code. Therefore, and from a theoretical point of view, the use of a single reply code is enough to detect an attacker trying to forge a successful vote delivery. Additionally, the use of only one reply code reduces approximately by $\frac{1}{3}$ the amount of information to be printed on the CodeCard.

It is also important to note that CodeVoting is vulnerable to malicious applications that change the correspondence between the candidates and the candidates' IDs. This vulnerability is due to the lack of secure output on the VST. However, there are measures to prevent ballot modification, such as i) publicly exposing the ballot some time before the election, ii) forcing the sorting of the candidates

on the ballot, and the corresponding candidate IDs, in a verifiable way, e.g. alphabetical sorting, and iii) using an image that is hard to forge/modify as a ballot.

4 Evaluation

We argue that CodeVoting protects against vote manipulations at the voter's PC under the following assumptions: i) the VST performs the protocols correctly and cannot be manipulated by the voter's computer, ii) the CodeCard is generated in a secure and controlled environment by the VST (the voter is the only person there), iii) the voter keeps her/his CodeCard secret, and iv) the correspondence between the candidate and its ID cannot be changed.

Under these assumptions, changing a vote to a predetermined candidate is virtually impossible because the corresponding vote code is not known by the attacker, and the *Probability of Guessing the Correct Vote Code* $P_{gvc} = \frac{1}{36^5}$, with 5 alphanumeric symbols, i.e. less than 1 in 60 millions. If we use both capital and non-capital letters, we have $P_{gvc} = \frac{1}{62^5}$, i.e. less than 1 in 900 millions.

A random change attack has almost the same probability of success as the previous attack. If we have n candidates, the *Probability of Guessing a Random valid Vote Code* is $P_{grvc} = n \cdot P_{gvc}$.

However, to prevent an easy denial of service attack the VST should not automatically block when the voter inserts invalid vote codes. Therefore, this limitation allows an attacker to perform a brute force attack to get a random valid vote code. Such attack can be minimized through simple measures, such as delaying the vote code verification function, e.g. with a delay of three seconds the attacker is limited to only 1200 tries in a one hour attack. Another possibility for reducing the chances of success of a brute force attack is increasing the domain of the vote codes, either by increasing the code length or by using more symbols, e.g. capital and non-capital letters. For instance, by using 62 symbols, the probability of a successful one hour brute force attack is 1 in 763444 for 5-digit codes, and 1 in more than 47 million for 6-digit codes.

The attacker can also try to manipulate the voter's vote by fooling the voter into believing that s/he has cast a vote, while in reality no vote was cast. To prevent such an attack we propose the use of a code to confirm the vote delivery. The VST only releases this code after getting a confirmation that the vote was successfully delivered, e.g. it could be a message signed by the election server. Therefore, if we use a confirmed vote delivery code with the same length as the vote codes, this attack has the same probability of success as the attack of changing a vote to a predetermined candidate. The confirmation received by the VST can be stored inside it to provide a proof of vote delivery, therefore allowing the voter to protest if her/his vote is not considered for the final tally.

Another important aspect is CodeVoting in the face of vote buying/coercion problems. CodeVoting allows the voter to produce a receipt of the vote by giving away the CodeCard to an attacker prior to the election day. On election day, the attacker can demand the voter to vote using a computer controlled by him/her,

e.g. by using a web site that is controlled by him/her or a special program also developed by the him/her. In this case, the attacker will have a vote receipt that proves the voter's vote, therefore enabling vote buying and coercion. We think the best way to prevent such an attack is by using a voting protocol that allows the voter to cast several votes, i.e. update her/his vote. We believe that the possibility of a vote update in a machine which is not controlled by the attacker would discourage vote buying and coercion attacks [8].

CodeVoting offers protection against vote manipulation and allows the detection of malicious interference in the voting process. However, CodeVoting cannot force a malicious computer to behave properly. Therefore, if any malicious interference in the voting process is detected, the voter should go to another computer and vote again.

4.1 Is the VST Trustworthy?

CodeVoting relies on the correct behavior of the VST. Therefore one can ask: CodeVoting is designed to protect the voter from the insecure voter's PC, but what guarantees are given that the VST does in fact do what it is supposed to do? One way verify it is by testing the VST. Besides testing the VSTs in the production phase, we believe that it would be good to have additional random testing by an independent certification authority.

Additionally, we can also make voters part of the certification process. It could be possible for a voter to verify her/his VST by running a fake election with instant results sometime before the real election.

Nevertheless, one can point out that the application running inside the VST is somehow able to detect that it is being subject to a test, and therefore will act properly in the tests but will still change the voter's vote on the real election day. This scenario can be prevented if one is sure of the software running inside the VST.

Fortunately, today there is secure tamper resistant hardware, such as smart cards, that supports signed applications. Therefore, it is possible to use publicly available and certified source code software. Of course, we can also have certified applications running on a PC. However, since it is possible for an attacker to take control of the voter's PC, a signed application does not guarantee the correct behavior. On the other hand, it is not possible to take control over secure tamper resistant hardware, or at least it should be very difficult even with specialized tools and impossible with just a common computer such as the ones at our homes. Therefore, a signed application running on secure tamper resistant hardware can guarantee the correct behavior.

5 Matrix CodeVoting

The presented CodeVoting technique has some limitations. First, it does not support large candidate lists, so its application is limited. Indeed, the CodeCard must have an entry for each possible candidate. If we consider elections with

a large number of candidates, let's say above thirty, the size of the CodeCard could start to become too large or usability problems may arise, e.g. due to small fonts used.

Another issue is the CodeCard reuse. If a voter uses the same CodeCard in more than one election, it is possible for an attacker to replace the voter's vote by a random one. To be able to do this, an attacker must collect the codes used during the first election. Additionally, the attacker must also be in control of the PCs used by the voter to vote in each election. If the voter uses different PCs, it will be much harder to perform the attack. We can also make this attack harder by executing the whole voting protocol inside the VST, without leaking the voter's identification to the voter's PC. Therefore, the different PCs only have the unlocking PIN to identify the voter.

Of course, the voter can always protect himself/herself against this type of attack by getting a new CodeCard for his/her VST between elections. However, an issue still remains: simultaneous elections. The simultaneous elections' issue is a particular case of the CodeCard reuse, in which the voter cannot go (or is not convenient for him/her to go) to the local authorities and get a new CodeCard. One solution that may work in some particular cases is the use of sequential candidate IDs throughout all the simultaneous elections, e.g. instead of using candidate IDs A and B for elections 1 and 2, use candidate IDs A and B for election 1 and candidate IDs C and D for election 2. This simple candidate numbering solves the problem of simultaneous elections but can lead to the large candidate list issue.

Next, we present the details of the Matrix CodeVoting, an enhancement to CodeVoting to support large candidate lists and, consequently, simultaneous elections as previously explained. Additionally, it also provides better security in the case of reuse of a CodeCard in consecutive elections. The Matrix CodeVoting allows the use of the CodeVoting technique in elections with a large candidate list by using an encryption matrix that stores a large number of candidate ID transformations in a compact form.

5.1 Matrix CodeVoting Details

The Matrix CodeVoting replaces the original CodeCard by a Matrix CodeCard (Fig. 4). The ballot format suffers minor changes and the candidates are identified by numbers (e.g. 54598, 39027, etc.) instead of letters (e.g. A, B, C, etc.), as illustrated in Fig. 5. The Matrix CodeCard consists in a matrix that the voter will use to translate (encrypt) the candidate ID to the vote code, in opposition to the direct associations printed in the original CodeCard.

Figure 6 shows how to use the Matrix CodeCard to get a vote code from a candidate ID. In this example we use a 5-digit candidate ID which supports 100000 different candidate IDs, a much larger number than the number of possible candidates with the first CodeCard design. With the help of the Matrix CodeCard the voter translates the candidate ID into a character string.

Of course, we still need to be careful when selecting the candidate IDs. For instance, if we have nine candidates it is a bad idea to use candidate IDs from

Matrix CodeCard							
Candidate number		d_5	d_4	d_3	d_2	d_1	
E n c o t d r i i n x g		0	A	S	Q	B	U
		1	W	E	P	S	I
		2	E	W	L	V	R
		3	R	Q	I	G	S
		4	Y	U	M	N	J
		5	V	N	H	Y	H
		6	B	J	T	U	T
		7	M	M	F	K	C
		8	O	X	E	W	E
		9	U	T	Z	A	P
Vote code		v_5	v_4	v_3	v_2	v_1	
Confirmed vote delivery code: 6HKG2							

Fig. 4. Matrix CodeCard for 5-digit candidate numbers

<p>Election for the most important figure in security.</p> <p>A - Alice B - Bob C - Eavesdropper D - Attacker</p> <p>Enter your vote code:</p>	<p>Election for the most important figure in security.</p> <p>54598 - Alice 39027 - Bob 78351 - Eavesdropper 46209 - Attacker</p> <p>Enter your vote code:</p>
--	--

Fig. 5. A regular ballot is presented on the left and the Matrix Code-Voting is shown on the right

Candidate number		4	6	2	0	9	
E n c o t d r i i n x g		0	A	S	Q	B	U
		1	W	E	P	S	I
		2	E	W	L	V	R
		3	R	Q	I	G	S
		4	Y	U	M	N	J
		5	V	N	H	Y	H
		6	B	J	T	U	T
		7	M	M	F	K	C
		8	O	X	E	W	E
		9	U	T	Z	A	P
Vote code		Y	J	L	B	P	

Fig. 6. Example of the encryption of a candidate ID using the Matrix CodeCard

00001 to 00009. In this case, the vote code for all candidates only differs in the last digit; therefore, an attacker willing to change the voter’s vote to another candidate would have a *Probability of Guessing a Valid Vote Code* $P_{gvc} = \frac{8}{25}$, i.e. 32% of success in the first try. This probability can be reduced by increasing the domain of the vote codes, e.g. by using capital and non-capital letters. With this example it is clear that the candidate IDs should be as different as possible to minimize the probability of an attacker guessing a valid vote code from the one typed in by the voter, e.g. 02536, 63875 and other IDs where all digits in the same positions are different.

Table 1. Number of different candidate IDs for $4 \leq k \leq 6$ and candidate ID length from 5 to 8. The results are an average of 100 rounds of random code generation.

Security Parameter	Candidate ID Length			
	5	6	7	8
$k = 4$	68	502	3483	25014
$k = 5$	-	65	387	2380
$k = 6$	-	-	58	298

Since the candidate IDs are in base 10 it is only possible to have 10 candidate IDs that differ in all positions. We propose the adoption of a security parameter k , which defines the number of different digits between all candidate IDs. The value of k must be chosen taking into account the probability of an attacker guessing, at the first try, a valid vote code from the one inserted by the voter. i.e. $P_{gvc} = (\frac{1}{\text{numberOfCodeSymbols}-1})^k$.

In Table 1 we present the possible number of candidates for $4 \leq k \leq 6$ and a candidate ID length from 5 to 8. Table 1 shows that the Matrix CodeVoting is able to securely support large candidate list elections by slightly increasing the length of candidate IDs.

As explained before, the problem of simultaneous elections can be reduced to a large candidate list problem. One hasty conclusion is to assume that if the Matrix CodeVoting offers a solution for elections with a large candidate list it can be used to securely conduct simultaneous elections. We must say that this is not entirely correct because if an attacker has access to different vote codes, generated with the help of the same Matrix CodeCard, it can correlate the codes, discover parts of the matrix and possibly substitute the voter’s vote.

The minimum number of codes needed to expose the entire matrix is 11, i.e. if a unique correspondence between the candidate numbers can be established only through an analysis of the repeated digits in the codes. For instance, given two simultaneous elections with the following pairs of candidates 12345, 67890 and 43567, 82059. A voter could vote for candidate 12345 in the first election and for candidate 82059 in the second election. Since the selected candidates are the only ones that share the second digit, the vote client application automatically has access to nine encrypted symbols in the encoding matrix, one in the column of the shared digit and two in all the other columns.

To minimize this threat it is important to choose candidate IDs in a way that no repeated digits in the vote codes of different elections can allow for a unique correspondence between candidate IDs, i.e. making the vote code correlation harder by adding confusion to the process.

Concluding, the Matrix CodeCard reuse should be limited to a few elections to prevent successful correlation attacks.

6 Conclusions and Future Work

Cryptographic voting protocols protect voting from manipulation at server side, if the protocol is executed by a trusted vote client. However, in remote voting, e.g. Internet voting, the voters use uncontrolled platforms to vote, therefore there are no guarantees that the client machine is trusted.

In this paper we presented the CodeVoting technique that protects the voter's vote from malicious manipulations at the uncontrolled/untrusted vote client platform, e.g. the voter's PC. CodeVoting uses a "code sheet" approach to protect the communication between a voter and a VST attached to the voter's PC. The VST runs a cryptographic voting protocol to protect the vote from manipulation at server side.

We presented the first code sheet format called CodeCard, that enables a secure communication between the voter and her/his VST. However, this format does not support elections with large candidate lists. Additionally, for improved security, the voter should renew her/his CodeCard between elections. Then, we introduced the Matrix CodeCard, a new CodeCard design that allows the support of large candidate lists. The use of the Matrix CodeCard in consecutive elections is also more secure.

In the future we want to enhance CodeVoting by introducing support for multi-candidate selection and ordering.

At the moment, we have working prototypes of the CodeCard and the Matrix CodeCard implemented in smart cards, namely JavaCards. We are now working on the integration of the CodeCard prototypes with a fully cryptographic vote client inside the VST (JavaCard).

Acknowledgments

We thank Patricia Lima and the editors of this book for the careful reading and suggestions made to improve the legibility of this manuscript.

References

1. CERT: Vulnerability remediation statistics (2007), http://www.cert.org/stats/vulnerability_remediation.html
2. USCERT: Cyber security bulletins (2007), <http://www.us-cert.gov/cas/bulletins/>

3. Wikipedia: Pharming (2007), <http://en.wikipedia.org/wiki/Pharming>
4. Stamm, S., Ramzan, Z., Jakobsson, M.: Drive-by pharming (2006), http://www.symantec.com/avcenter/reference/Driveby_Pharming.pdf
5. Gaudin, S.: Pharming attack slams 65 financial targets. InformationWeek (2007), <http://www.informationweek.com/showArticle.jhtml?articleID=197008230>
6. Kirk, J.: Pharming attack hits 50 banks. IDG News Service, TechWorld (2007), <http://www.techworld.com/security/news/index.cfm?newsid=8102>
7. Council of Europe: Family voting. Congress of Local and Regional Authorities of Europe session (2002), http://www.coe.int/T/E/Com/Files/CLRAE-Sessions/2002-06-Session/family_voting.asp
8. Volkamer, M., Grimm, R.: Multiple casts in online voting: Analyzing chances. In: Robert Krimmer, R. (ed.) Electronic Voting 2006, Castle Hofen, Bregenz, Austria. LNI, vol. P-86, pp. 97–106. GI (2006)
9. California Internet Task Force: A report on the feasibility of internet voting (2000), <http://www.ss.ca.gov/executive/ivote>
10. Internet Policy Institute: Report of the national workshop on internet voting: Issues and research agenda (2001), <http://www.diggo.org/archive/library/dgo2000/dir/PDF/vote.pdf>
11. Jefferson, D., Rubin, A.D., Simons, B., Wagner, D.: A security analysis of the secure electronic registration and voting experiment (serve) (2004), <http://www.servesecurityreport.org/paper.pdf>
12. Rivest, R.L.: Electronic voting. In: Syverson, P.F. (ed.) FC 2001. LNCS, vol. 2339, p. 243. Springer, Heidelberg (2001)
13. Rubin, A.D.: Security considerations for remote electronic voting. Commun. ACM 45(12), 39–44 (2002)
14. Joaquim, R., Ribeiro, C.: Codevoting: protecting against malicious vote manipulation at the voter's pc. In: Chaum, D., Kutylowski, M., Rivest, R.L., Ryan, P.Y.A. (eds.) Frontiers of Electronic Voting, no. 07311 in Dagstuhl, Germany. Dagstuhl Seminar Proceedings, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany (2007)
15. Joaquim, R., Ribeiro, C.: CodeVoting protection against automatic vote manipulation in an uncontrolled environment. In: Alkassar, A., Volkamer, M. (eds.) VOTE-ID 2007. LNCS, vol. 4896, pp. 178–188. Springer, Heidelberg (2007)
16. Fujioka, A., Okamoto, T., Ohta, K.: A practical secret voting scheme for large scale elections. In: Zheng, Y., Seberry, J. (eds.) AUSCRYPT 1992. LNCS, vol. 718, pp. 244–251. Springer, Heidelberg (1993)
17. Joaquim, R., Zúquete, A., Ferreira, P.: Revs - a robust electronic voting system (extended). IADIS International Journal of WWW/Internet 1(2), 47–63 (2003)
18. Ohkubo, M., Miura, F., Abe, M., Fujioka, A., Okamoto, T.: An improvement on a practical secret voting scheme. In: Zheng, Y., Mambo, M. (eds.) ISW 1999. LNCS, vol. 1729, pp. 225–234. Springer, Heidelberg (1999)
19. Okamoto, T.: Receipt-free electronic voting schemes for large scale elections. In: Christianson, B., Crispo, B., Lomas, M., Roe, M. (eds.) Security Protocols 1997. LNCS, vol. 1361, pp. 25–35. Springer, Heidelberg (1998)
20. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM 24(2), 84–88 (1981)

21. Clarkson, M., Chong, S., Myers, A.: Civitas: A secure remote voting system. In: Chaum, D., Kutylowski, M., Rivest, R.L., Ryan, P.Y.A. (eds.) *Frontiers of Electronic Voting*, Dagstuhl, Germany. Dagstuhl no. 07311 in Seminar Proceedings, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany (2007)
22. Neff, C.A.: Verifiable mixing (shuffling) of elgamal pairs (2004), <http://votehere.com/vhti/documentation/egshuf-2.0.3638.pdf>
23. Park, C.-s., Itoh, K., Kurosawa, K.: Efficient anonymous channel and all/Nothing election scheme. In: Helleseth, T. (ed.) *EUROCRYPT 1993*. LNCS, vol. 765, pp. 248–259. Springer, Heidelberg (1994)
24. Benaloh, J.C.: *Verifiable Secret-Ballot Elections*. PhD thesis, Yale University (1987)
25. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: Fumy, W. (ed.) *EUROCRYPT 1997*. LNCS, vol. 1233, pp. 103–118. Springer, Heidelberg (1997)
26. Damgård, I., Jurik, M.: A generalisation, a simplification and some applications of paillier’s probabilistic public-key system. In: Kim, K.-c. (ed.) *PKC 2001*. LNCS, vol. 1992, pp. 119–136. Springer, Heidelberg (2001)
27. Hirt, M., Sako, K.: Efficient receipt-free voting based on homomorphic encryption. In: Preneel, B. (ed.) *EUROCRYPT 2000*. LNCS, vol. 1807, pp. 539–556. Springer, Heidelberg (2000)
28. Estonian National Electoral Committee: *Internet voting in estonia* (2007), <http://www.vvk.ee/engindex.html>
29. Lee, B., Kim, K.: Receipt-free electronic voting scheme with a tamper-resistant randomizer. In: Lee, P.J., Lim, C.H. (eds.) *ICISC 2002*. LNCS, vol. 2587, pp. 389–406. Springer, Heidelberg (2003)
30. Oppliger, R.: How to address the secure platform problem for remote internet voting. In: Erasim, E., Karagiannis, D. (eds.) *5th Conference on “Sicherheit in Informationssystemen” (SIS 2002)*, Vienna, Austria, pp. 153–173. vdf Hochschulverlag (2002)
31. Zúquete, A., Costa, C., Rom ao, M.: An intrusion-tolerant e-voting client system. In: *1st Workshop on Recent Advances on Intrusion-Tolerant Systems (WRAITS 2007)*, Lisbon, Portugal (2007)
32. TGC: Trusted computing group (2007), <https://www.trustedcomputinggroup.org/home>
33. Sadeghi, A.R., Selhorst, M., Stübke, C., Wachsmann, C., Winandy, M.: Tcg inside?: a note on tpm specification compliance. In: *STC 2006: Proceedings of the first ACM workshop on Scalable trusted computing*, Alexandria, Virginia, USA, pp. 47–56. ACM, New York (2006)
34. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation. In: Pfitzmann, B., Liu, P. (eds.) *CCS 2004: Proceedings of the 11th ACM conference on Computer and Communications Security*, Washington DC, USA, pp. 132–145. ACM, New York (2004)
35. Volkamer, M., Alkassar, A., Sadeghi, A.R., Schulz, S.: Enabling the application of the open systems like pcs for online voting. In: *Frontiers in Electronic Elections Workshop (FEE 2006)*, Hamburg, Germany (2006)
36. Chaum, D.: Surevote (2001) International patent WO 01/55940 A1, <http://www.surevote.com/home.html>
37. UK’s Electoral Commission: *Technical report on the may 2003 pilots* (2003), <http://www.electoralcommission.org.uk/about-us/03pilotscheme.cfm>

38. UK's National Technical Authority for Information Assurance: e-voting security study (2002),
http://www.ictparliament.org/CDTunisi/ict_compendium/paes/uk/uk54.pdf
39. Helbach, J., Schwenk, J.: Secure internet voting with code sheets. In: Alkassar, A., Volkamer, M. (eds.) VOTE-ID 2007. LNCS, vol. 4896, pp. 166–177. Springer, Heidelberg (2007)
40. Kutylowski, M., Zagórski, F.: Verifiable internet voting solving secure platform problem. In: Miyaji, A., Kikuchi, H., Rannenberg, K. (eds.) IWSEC 2007. LNCS, vol. 4752, pp. 199–213. Springer, Heidelberg (2007)
41. Skagestein, G., Haug, A.V., Nødtvedt, E., Rossebø, J.E.Y.: How to create trust in electronic voting over an untrusted platform. In: Krimmer, R. (ed.) Electronic Voting 2006, Castle Hofen, Bregenz, Austria. LNI, vol. P-86, pp. 107–116. GI (2006)

A Practical and Secure Coercion-Resistant Scheme for Internet Voting

(Extended Abstract)

Roberto Araújo¹, Sébastien Foulle², and Jacques Traoré²

¹ TU-Darmstadt, Hochschulstrasse 10, 64289 Darmstadt, Germany
`rsa@cdc.informatik.tu-darmstadt.de`

² Orange Labs, 42 rue des Coutures, BP 6243, 14066 Caen Cedex, France
`s.foulle.ext@rd.francetelecom.com`, `jacques.traore@orange-ftgroup.com`

Abstract. Juels, Catalano, and Jakobsson (JCJ) proposed at WPES 2005 the first voting scheme that considers real-world threats and that is more realistic for Internet elections. Their scheme, though, has a quadratic work factor and thereby is not efficient for large scale elections. Based on the work of JCJ, Smith proposed an efficient scheme that has a linear work factor. In this paper we first show that Smith's scheme is insecure. Then we present a new coercion-resistant election scheme with a linear work factor that overcomes the flaw of Smith's proposal. Our solution is based on the group signature scheme of Camenisch and Lysyanskaya (Crypto 2004).

1 Introduction

Remote electronic elections may provide many benefits to democratic societies. They may increase elections turnouts, afford convenience to the voters, and reduce costs, for instance. The risks inherent in such elections, though, can discourage its use in major political elections. The main threat is that coercion and vote-selling can be easily explored by adversaries. Remote elections, thus, must have ways to prevent or at least mitigate such problems.

Most existing proposals for remote elections rely on the receipt-freeness requirement and on assumptions to deal with coercion and vote-selling. Preventing receipts to be made, though, is not enough to counter these problems as the voter can be observed while voting, for example. Many assumptions (e.g. the voter cannot give away her private key material) are unrealistic for remote scenarios.

Recently, Juels, Catalano, and Jakobsson (JCJ) [17] introduced a more complete requirement for remote elections called coercion-resistance. This concept considers not only the receipt-freeness requirement, but also real world attacks related to coercion (and vote-selling). Coercion-resistance takes into account that an adversary can force the voter to abstain from voting, can obtain private information from the voter and vote on her behalf, or can force the voter to send a randomly formed ballot as her vote.

Besides the coercion-resistance requirement, JCJ introduced the first scheme that fulfills it. The scheme basically mitigates coercive attacks by allowing the voter to deceive adversaries about her vote intention. It, though, requires a quadratic work factor (in number of votes) to compute the voting results and hence it is impractical for large scale elections. Particularly, the scheme relies on an inefficient blind comparison mechanism to determine the results.

Based on the JCJ solution, Smith [24] presented an efficient coercion-resistant scheme. The proposal replaces the comparison mechanism of JCJ by a new one that computes the voting results in a linear time. Also, it corrects some problems observed by Smith in JCJ scheme. Weber et al. [26], however, pointed out problems of Smith's proposal and presented a protocol that combines the ideas of JCJ with a variant of Smith's mechanism.

Paper Contribution and Organization

In this work we first present a weakness in Smith's mechanism of comparison that makes his scheme insecure in the sense of coercion-resistance. The problem is also relevant to the scheme of Weber et al. as it employs the ideas of Smith. We introduce a new coercion-resistant scheme with a linear work factor. The solution is based on JCJ ideas, but it has a linear work factor and does not rely on inefficient comparisons to compute the voting results.

The paper is organized as follows: in Section 2 we review the proposal of JCJ and the comparison mechanism of Smith; also, we describe the weakness of Smith's solution. After that, in Section 3, we present our proposal of coercion-resistant scheme. Then, we sketch an analysis of our proposal in Section 4. Finally, we conclude our work in Section 5.

2 The Scheme of JCJ and Smith's Comparison Mechanism

In this section we recall shortly the proposal of Juels, Catalano, and Jakobsson (JCJ) and present the mechanism of comparison proposed by Smith. Also, we show the weakness of Smith's mechanism.

2.1 The Proposal of JCJ

The scheme of Juels, Catalano, and Jakobsson [17] relies essentially on a method of indirect identification through anonymous credentials to overcome coercive attacks. Especially, the voter receives a valid credential (e.g. an alphanumeric string) in a secure way and uses it when she want to cast her valid vote. A voter under coercion, though, is able to make a fake credential and to hand it to a coercer. After the end of the voting, a blind comparison mechanism distinguishes the valid credentials from the fake ones to identify the valid votes; conversely, an adversary has no way to perform this distinction.

The scheme considers a registration phase free of adversaries and a bulletin board communication model. Also, it requires the following cryptographic tools: non-interactive zero-knowledge proofs, a probabilistic threshold public-key cryptosystem, and universally verifiable mix nets. In particular, the scheme employs a plaintext equivalence test [15]. This primitive takes two ciphertexts as input and returns a bit indicating if the corresponding plaintexts are equal or not. The JCJ solution is briefly described as follows:

Registration phase. In this phase a trustworthy authority issues a unique valid credential, which is a random value, for each eligible voter and publishes a probabilistic encryption of each credential on the bulletin board. Let $L1$ be the list containing all credential ciphertexts published by the authority on the bulletin board.

Voting phase. In order to vote, a voter sends the following data to the bulletin board through an anonymous channel: a tuple containing her encrypted vote, her encrypted credential, and zero-knowledge proofs that the vote is for a valid candidate and that the voter knows the vote and the credential encrypted.

Tallying phase. At the end of the voting, the talliers verify all proofs posted on the board and exclude tuples with invalid proofs. From the remaining tuples, they perform a pairwise blind comparison by means of the plaintext equivalence test to remove tuples with duplicated credentials. After removing the duplicates keeping the last posted tuples, the remaining pairs of ciphertexts (a vote and a credential) form the list $L2$ and this list is sent to a mix net. The mix net returns $L2'$. Then, the list $L1$ created during the registration phase is sent to a different mix net that returns $L1'$. Now, the plaintext equivalence test is used a second time to compare (pairwise) the credentials on the list $L1'$ with the credentials on the list $L2'$. A vote is removed if its encrypted credential in $L2'$ does not match with an element of $L1'$. Finally, the votes with valid credentials are decrypted by the talliers.

Drawback. Although the JCJ scheme fulfills the coercion-resistance requirement, the pairwise blind comparisons involving the plaintext equivalence tests makes it inefficient for large scale elections. Let N be the number of voters and V be the number of posted votes, one has $V \geq N$ and the overhead to perform the tests is quadratic in V .

2.2 Smith's Comparison Mechanism

Based on the JCJ proposal, Smith [24] introduced a more efficient coercion-resistant scheme. The solution substitutes the previous comparison mechanism of JCJ for a new one that computes the voting results in a linear time. Moreover, it includes a timestamp in the tuple that the voter submits and a further mix step in the tallying phase. The latter improvements, though, are irrelevant as pointed out by Weber et al. [26].

The mechanism of Smith performs a global blind comparison of ciphertexts instead of pairwise comparing ciphertexts via a plaintext equivalence test. In order to accomplish this, the method makes deterministic fingerprints from probabilistic encrypted credentials and then compares the resulting fingerprints through hash tables. The method depends on the El Gamal cryptosystem and is described as follows:

Let s be an El Gamal private key shared among the talliers and corresponding to a public key $h = g^s$, where g is a group generator, k another private key shared, ks the product of k and s also shared, and $(g^r, \sigma h^r)$ the El Gamal ciphertext of a credential σ , where r is a random number. In order to make a fingerprint from $(g^r, \sigma h^r)$, the talliers cooperatively compute $(g^r)^{ks} = h^{rk}$ and $(\sigma h^r)^k = \sigma^k h^{rk}$. Then, they divide $(\sigma^k h^{rk})$ by (h^{rk}) to obtain σ^k . The talliers now use half of the bits of σ^k as the fingerprint. This process is applied to all credential ciphertexts using the same k and ks before comparing the resulting fingerprints.

Observe that the talliers need to publish σ^k before making the fingerprint. Thus, anyone can verify the fingerprint is correct.

Weakness. Smith's comparison method is efficient. However, it is insecure. Especially, an adversary can determine whether a coerced voter gave him a valid or a fake credential [4]. In order to show this, we consider the following scenario:

Suppose an adversary forces the voter to reveal her credential σ . Now, the adversary makes two tuples, one with the encryption of σ and the other with the encryption of σ^2 , and publishes them on the bulletin board. In the tallying phase, after applying Smith's method, the talliers publish σ^k and σ^{2k} on the board. Now, by squaring a copy of each element on the board, the adversary is able to test if a squared element matches an element on the board. Thus, if the two votes corresponding to σ and its square were removed by the talliers, the coercer learns that σ is an invalid credential.

3 Our Coercion-Resistant Voting Scheme

As we presented before, the scheme of JCJ is inefficient for large scale elections. Also, we showed that the comparison mechanism of Smith is insecure. We now introduce a new coercion-resistant voting scheme that employs some of the JCJ ideas and that computes voting results in a linear time.

Our solution does not rely on blind comparisons to identify valid credentials. Instead, we employ a particular mathematical structure to make the credentials and use a function to identify them apart. The structure makes hard for a coercer or a dishonest voter to forge new valid credentials, even after having seen several valid ones.

The new scheme has the following advantages: its security can be proved, it is a practical linear scheme (in the number of votes posted by the voters), one cannot link the votes of a given voter in different elections, and the generation of the credentials as well as the verification of their validity can be distributed

¹ This problem was also observed independently by Clarkson et al. [8].

among several authorities. Thus, a single corrupted authority cannot give valid credentials to an attacker or tell to a coercer whether a credential is valid or not.

3.1 Building Blocks

The scheme requires the following tools:

Bulletin Boards. In order to help achieving global verifiability, our proposal relies on bulletin boards as communication model. In this model the bulletin board performs as a public broadcast channel by receiving information and by allowing anyone to read any information received. Also, once receiving information, the board stores it and cannot delete or modify it.

Bulletin boards may be implemented via a Byzantine agreement, such as the proposal of Cachin et al. [4].

Universally Verifiable Mix Nets. In some steps of our scheme we employ mix nets to provide anonymity. This cryptographic primitive was introduced by Chaum [6] and further developed by many other authors. It performs by permuting messages, and by reencrypting or by decrypting them. The scheme requires a re-encryption mix net based on the El Gamal cryptosystem as introduced by Park et al. [22]. However, in order to reduce the trust in the mix process, the mix net should be universally verifiable. That is, after mixing messages, the mix net must prove publicly the correctness of its shuffle. The proposal of Neff [19] is an example of a universally verifiable mix net.

A Threshold Cryptosystem. Our scheme relies on a threshold version of a semantically secure cryptosystem with homomorphic property, such as Paillier [21] or El Gamal [12] under secure groups. We require here, though, the Modified El Gamal cryptosystem proposed by JCJ [17]. This variant is described as follows: let G be a cyclic group of order p where the decision Diffie-Hellman problem (see Boneh [1] for details) is hard, the public key is composed of the elements $(g_1, g_2, h = g_1^{x_1} g_2^{x_2})$ with $h, g_1, g_2 \in G$ and the corresponding private key is formed by $x_1, x_2 \in Z_p$. The Modified El Gamal ciphertext of a message $m \in G$ is $(M = g_1^s, N = g_2^s, O = mh^s)$, where $s \in Z_p$ is a random number. The message m is obtained from the ciphertext (M, N, O) by $O/(M^{x_1} N^{x_2})$. In the threshold version, the El Gamal public key and its corresponding private key are cooperatively generated by n parties; though, the private key is shared among the parties. In order to decrypt a ciphertext, a minimal number of t out of n parties is necessary. See Cramer et al. [9] for a description of an El Gamal threshold cryptosystem and Gennaro et al. [13] for a secure key generation protocol.

Non-Interactive Zero-knowledge Proofs. The proposal we present below also requires several zero-knowledge proof protocols. These primitives help ensuring security in our solution. The scheme employs Schnorr signatures [23] to make ciphertexts plaintext aware (i.e. the party who makes the ciphertext should be aware of what he is encrypting) and so preventing the use of the El Gamal

malleability by adversaries; in addition, the verifier should check that the components of the ciphertexts are of order p to prevent the attacks described in [18]. The solution, moreover, requires a protocol to prove that a ciphertext contains a vote for a valid candidate. This can be accomplished, for example, through the validity proof proposed by Hirt and Sako [14]. Besides these protocols, our proposal uses the discrete log equality test owing to Chaum and Pedersen [7], a protocol for proving knowledge of a representation, such as the one proposed by Okamoto [20], and a plaintext equivalence test [15]. We employ the Fiat-Shamir heuristic [10] to convert interactive zero-knowledge proofs into non-interactive ones.

3.2 Security Model

The coercion-resistance requirement takes into account typical problems of remote votings. The scheme we introduce here aims at satisfying this requirement. In order to fulfill it, however, our proposal depends on certain assumptions and conditions described here. Similarly to JCJ scheme, we consider the following:

- An adversary may impede the voter to vote, force the voter to post a random composed ballot material, or demands secret information from the voter. The adversary has limited computational power and is able to compromise only a small number of authorities;
- The adversary may monitor the voter or interact with her during the voting process. However, we suppose the adversary is not able to monitor or interact with the voter continuously during the voting period;
- A registration phase free of adversaries. Also, we assume that the voter communicates to the registrars via a untappable channel and without the interference of adversaries. This channel provides information-theoretical secrecy to the communication;
- Some anonymous channels in the voting phase. The voters are supposed to have access to these channels and use them to post their votes;
- Voters cast their votes by means of reliable machines.

3.3 The Credentials

In the proposals of JCJ and of Smith, a valid credential is a random string. Here, differently, a credential has a mathematical structure based on the group signature scheme of Camenisch and Lysyanskaya [5].

A valid credential in our scheme has the following form: let G be a cyclic group with prime order p where the decision Diffie-Hellman (DDH) problem is hard, (x, y) two secret keys, a a random number in G (with $a \neq 1$), r a random number in Z_p , the credential is composed of $(r, a, b = a^y, c = a^{x+ry})$.

The security of our credentials relies heavily on the LRSW (see [5] for details) and on the DDH assumptions. The former assumption ensures that even if an adversary has many genuine credentials (r_i, a_i, b_i, c_i) , it is hard for him to forge a new and valid credential (r, a, b, c) , with $r \neq r_i$ for all i . This assumption is known

to hold for generic groups and the security of Camenisch and Lysyanskaya's signature scheme also relies on it. The DDH assumption ensures that the voter cannot prove to anyone else whether (r, a, b, c) is a valid credential or not. This way, a voter under coercion can make a fake r to deceive an adversary who will not be able to distinguish between a fake and a valid r .

In a real-world scenario, our credential can be seen as containing two parts: a short one, that is r , which must be kept secret, and a long one, that is (a, b, c) . The first part (i.e. r) has around twenty ASCII characters (this corresponds to 160 bits, the actual secure size for the order of generic groups), so a small piece of paper and a pen are sufficient to write r down. The other part can be stored in a device or be even sent by email to the voter without compromising the credential security.

3.4 The Scheme

Taking into account the security model and the building blocks introduced as well as the new credential, we present now our coercion-resistant voting scheme. The proposal is composed of four phases, that is, setup, registration, voting, and tallying, and involves three participants:

Voter. The voter is denoted by V . She holds a valid credential and uses it when she wants to cast her valid vote. Also, she is able to make fake credentials and use them to deceive adversaries;

Talliers. These authorities are represented by T . They are responsible for controlling the bulletin board, for running the mix net, and for computing the voting results. They share a Modified El Gamal private key \hat{T} corresponding to a public key T ;

Registrars. They are denoted by R . These authorities are responsible for issuing a valid credential for each eligible voter. Also, they help the talliers to identify valid credentials. They share two private keys, x and y corresponding to the public keys R_x and R_y .

We employ the following notation in the description below: BB is a bulletin board, $E_T[m]$ is a Modified El Gamal encryption of a message m constructed with T , and $D_{\hat{T}}[m]$ is a Modified El Gamal decryption of m .

The scheme consists of the following procedures:

Setup phase. In this phase the general voting parameters are established and published along with a digital signature on BB . These parameters consist of a cyclic group G with prime order p where the decision Diffie-Hellman problem is hard, a random generator o of G as well as the keys of T . Especially, the talliers T cooperate to generate the public key T and the shared private key \hat{T} via the Modified El Gamal threshold cryptosystem. The registrars R collaborate to produce their public keys R_x and R_y and their respective shared private keys x and y ; these public keys are computed as follows: $R_x = g^x$ and $R_y = g^y$ where g is a public random generator of G . Also, the list of voting candidates available is published.

Registration phase. After verifying that a voter is eligible, R issues to the voter a secret credential $\sigma = (r, a, b, c)$ via an untappable channel, where a is a random element in G (with $a \neq 1$), r is a random element in Z_p , $b = a^y$, and $c = a^{x+rx y}$. In addition, R may furnish the voter with a designated verifier proof [16] of well-formedness for σ . Note that if (r, a, b, c) is valid, then for all r the credential (r, a^l, b^l, c^l) for $l \in_R Z_p$ is a valid one too. This property is used by the voter to change the values a, b, c each time she votes.

Voting phase. The voter casts her ballot by sending the tuple $(E_T[C], a, E_T[a^r], E_T[a^{ry}], E_T[a^{x+rx y}], o^r, P)$ through an anonymous channel to BB , where C is the candidate chosen, $(r, a, a^r, a^{ry}, a^{x+rx y})$ correspond to voter's credential, o is the public generator of G published in the setup phase, and P is a list of non-interactive zero-knowledge proofs which ensure that the vote is well-formed. In particular, P contains a proof that the vote is for a valid candidate, proofs of knowledge of the plaintexts, and a proof that $E_T[a^r]$ and o^r contain the same r . Recall from the previous paragraph that the values $a, b = a^y$, and $c = a^{x+rx y}$ have been changed by the voter and are therefore different from the ones she received from R .

The value o^r is used to detect duplicates and guarantees that only one vote per voter will be counted. Otherwise, a dishonest voter could vote several times without being detected.

Tallying phase. In order to compute the voting results, the talliers T perform the following steps:

1. **Verifying proofs.** T verifies the proofs P on each tuple and remove tuples with invalid proofs. That is, T verifies that a is in G and $a \neq 1$, that $E_T[C]$ is a vote for a valid candidate, the proofs of knowledge of the plaintexts, and the proof that $E_T[a^r]$ and o^r contain the same r ;
2. **Removing duplicates.** In order to exclude duplicates, T first identifies them by comparing all o^r , for instance, using a hashtable. After this, T keeps the last posted tuples based on the order of posting on the bulletin board;
3. **Encrypting the plaintext element.** The tuples that passed the previous steps have their values o^r and P deleted, and their second component (i.e. a) replaced by the Modified El Gamal ciphertext $E_T[a]$. This way, only the values $E_T[C], E_T[a], E_T[a^r], E_T[a^{ry}], E_T[a^{x+rx y}]$ are processed in the next step;
4. **Mixing tuples.** T sends the tuples composed of $E_T[C], E_T[a], E_T[a^r], E_T[a^{ry}], E_T[a^{x+rx y}]$ to a verifiable mix net and publish the output on BB . Let the tuples formed by $(t, u, v, w, z) = (E_T[C]', E_T[a]', E_T[a^r]', E_T[a^{ry}]', E_T[a^{x+rx y}]')$ be the mix net output, where $E_T[X]'$ means a re-encryption of $E_T[X]$;
5. **Identifying valid votes.** For each tuple, R first employs its secret key y to cooperatively compute v^y . Then, R checks whether v^y and w have the same plaintext using a plaintext equivalence test. If the verification result is positive, R generates a fresh shared key $\alpha \in_R Z_p$ and cooperatively computes

$(zu^{-x}w^{-x})^\alpha$ using the shared private key x that was generated along with y in the setup phase. Now T collaborates to decrypt the resulting ciphertext processed by R . The decryption is equal to 1 if and only if the credential is a valid one. Note that if the credential is invalid, just computing and decrypting $(zu^{-x}w^{-x})$ may give some information to an adversary, so the random exponent α is necessary.

6. **Decrypting and counting the votes.** T employs its shared private key \widehat{T} to cooperatively decrypt $E_T[C]$ of each tuple with a valid credential. After that, they count the votes and publish the results on BB .

Notice that a voter under coercion should reveal the correct values a and b of her credential. Otherwise, an adversary can test whether this pair is correct by mounting a "1009 attack" [24]. That is, the adversary sends "1009" ballots containing pairs of the form (a^{l_i}, b^{l_i}) using 1009 random values l_i and checks whether more than 1009 ballots passed the first test in step 5 of the tallying phase.

3.5 Multiple Elections

The number of eligible voters may change in different elections. Some voters may have their right to vote revoked after having participated in an election, for instance. Also, a voter may be allowed to vote in several elections, but may not vote in others. In order to satisfy these scenarios, a credential is normally required to be used in multiple elections and should be revoked by the authorities when necessary.

The credential we proposed may be used in multiple elections as long as the same keys (x, y) are employed. However, in principle a credential cannot be revoked. As only the voters knows their credentials, the authorities are not able to revoke a credential. In addition, even if the authorities store all credentials issued, they are not able to efficiently identify a revoked credential since the credentials are published in an encrypted form.

Although the design of our scheme makes revocation difficult, the scheme has some properties that help accomplishing this. Upon registering, a voter receives $(r, a, b = a^y, c = a^{x+rx})$. As stated before, the element r must be transmitted via an untappable channel. However, the elements $(a, b = a^y, c = a^{x+rx})$ may be sent by post or even by email; this does not compromise the credential security as long as the DDH assumption holds. Based on this, we suggest the following method to revoke credentials and to perform new elections:

Besides generating and issuing a credential for each voter, the registrars R cooperatively compute the encryption of (a^r) and (a) (i.e. $E_R[a], E_R[a^r]$) and stores them in a list. These encryptions are performed using a public key R corresponding to a shared private key \widehat{R} especially generated for this purpose.

For each new election, instead of using the same keys (x, y) , the registrars generates new keys (x', y') and furnish the voters with new values $(a' = a^l, b' = a^{ly'}, c' = a^{l(x'+rx'y')})$, computed from $E_R[a]$ and $E_R[a^r]$, for a randomly chosen l . That is, c' is computed by raising $E_R[a]$ and $E_R[a^r]$ to x' and to $x'y'$ respectively, and then by using homomorphism to obtain $E_R[a^{x'+rx'y'}]$. After that,

$E_R[a^{x'+rx'y'}]$ is raised to l and cooperatively decrypted. The values a' and b' can be obtained similarly, but without using homomorphism. The new elements of the credential could be sent by mail to the voter or published on a dedicated website.

4 Analysis

The scheme presented in the previous section aims at fulfilling the coercion-resistant requirement as well as standard voting security requirements. We sketch here an analysis of our scheme based on these requirements and considering the security model introduced before.

4.1 Coercion Resistance

In order to be coercion resistant, a voting scheme must be receipt-free and defeat coercive attacks, such as randomization, forced-abstention, and simulation attacks, as defined by JCJ.

A scheme is receipt-free if the voter is not able to make or obtain a receipt to prove in which way she has voted. Especially, the voter here may not convince an adversary that her credential is valid and that she used it to cast a particular vote. Our proposal satisfies these requisites. The voter is not able to prove an adversary that her credential is valid and an adversary cannot determine whether a credential is valid or not unless he can break the DDH problem. In addition, the credentials are verified only after a mixing process and the method employed to verify them (see step 5 in the tallying phase) does not leak any information. This way, the voter is not able to obtain any evidence that can be used as a proof.

The proposal we presented is resistant to the randomization attack as well. In this attack an adversary forces the voter to cast a ballot composed of random information. As the voter in our scheme publishes her vote along with a set of zero-knowledge proofs and all votes with invalid proves are excluded, ballots randomly composed will not be tallied. In addition, even if the adversary observes the voter and forces her to vote for a random candidate, she cannot verify the voter performed this using her valid credential.

In the forced-abstention attack an adversary forces the voter to abstain from voting. This attack is possible if the adversary can verify the voter has voted. Our scheme, however, does not reveal any information about the voter identity. The voter receives a valid credential that identifies her, but it is kept hidden from adversaries. That is, the voter publishes the credential ciphertext on the bulletin board via an anonymous channel and the credential is verified in the tallying phase (step 5) without being decrypted. Hence, the adversary cannot check whether the voter has voted or not.

The fact that the voter's identity is concealed also prevent an adversary from forcing a voter to show the random exponents used for encrypting her ballot components. As the voter posts her ballot through an anonymous channel and

no information about the credential is revealed during the tallying, the adversary does not know who voted. This way, a coerced voter can say an adversary that she did not vote and he cannot verify whether the voter told him the truth or not. An adversary could also force the voter to reveal the exponents before she sends her ciphertexts. However, the voter can use a fake credential and show the exponents of the corresponding components.

Our scheme also prevents the simulation attack. In this attack an adversary forces the voter to reveal her valid credential and vote on her behalf. However, the voter in our solution is able to deceive the adversary by handing him a fake credential and the adversary cannot distinguish a valid credential from a fake one under the DDH assumption. The credential structure, the mix process as well as the method used to identify valid credentials avoid the adversary performing the distinction.

4.2 Democracy and Accuracy

In our proposal, the bulletin board may accept votes from eligible and non-eligible voters and the voters may vote multiple times. However, only votes from eligible voters appear in the final tally and only one vote per eligible voter is counted. The scheme accomplishes this by excluding votes posted with the same credential (see step 3 in tallying phase). This way, even if a voter uses the same credential to vote many times, only the last vote will be processed. In addition, the scheme checks whether the credentials are valid or not and excludes votes with fake credentials. This is performed by the method that identifies valid credentials in step 5 of the tallying phase. Since the method only outputs the value one for valid credentials and that it is hard to forge valid credentials under the LRSW assumption, it ensures that only votes from eligible voters will be in the final tally. Conversely, the method outputs a random number as result for invalid credentials. This way, votes from non-eligible voters (i.e. invalid votes) will not be counted.

4.3 Universal Verifiability

Anyone is able to verify the correctness of the voting process and its results in our solution. This requirement is ensured by the public bulletin board which is secure and by the non-interactive zero-knowledge proofs (NIZKPs). The proofs generated in all phases of the scheme are published on the bulletin board to allow anyone to verify them. In addition, the voters publish their votes on bulletin board, so anyone is able to verify the votes that will be processed. In the tallying phase, the steps performed can also be verified by anyone through the bulletin board; this includes the shuffle performed by the mix net and our method to identify valid credentials.

The bulletin board and the NIZKPs also prevent the disassociation of the pair of ciphertexts (a vote and a credential). After the voter publishes her ballot on the board, any transformation of the ciphertexts (i.e. re-encryptions) is proved through the NIZKPs.

4.4 Efficiency

As stated before, the JCJ scheme requires a quadratic running time. The reason for this is the pairwise blind comparison mechanism used for removing duplicates and for identifying valid credentials. Our proposal, differently, does not rely on blind comparisons. The duplicates are identified in the scheme by comparisons that can be performed in a linear time, for instance by means of a hash table. Similarly, the scheme identifies valid credentials by testing each credential apart and this can be also performed efficiently. Thus, let N be the number of eligible voters and V the number of posted votes, our scheme has a running time $O(N+V)$. As V may be much bigger than N , our scheme is linear in the number of votes.

5 Conclusion

The scheme of Juels, Catalano, and Jakobsson (JCJ) considers realistic threats and is more suitable for Internet elections. Unfortunately their scheme is inefficient for large scale elections. Smith proposed an improved scheme, but his solution is not coercion-resistant as we showed.

We have introduced a practical and secure scheme that satisfies the property of coercion-resistance. Our scheme inherits some ideas from the JCJ protocol as the use of anonymous credentials. It, however, employs special credentials of which security depends on the DDH and on the LRSW assumptions; moreover, it does not rely on comparisons to identify valid credentials and is efficient for large scale elections.

The solution presented is based on the group signature scheme of Camenish and Lysyanskaya. We have a variant of our proposal that employs the protocol of Boneh et al. [2]. This variant will be presented in a forthcoming paper.

We have argued that our scheme is secure, but have not formally proved this property. We will provide a formal proof in the full version of this paper.

References

1. Boneh, D.: The decision diffie-hellman problem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 48–63. Springer, Heidelberg (1998)
2. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin [11], pp. 41–55
3. Brickell, E.F. (ed.): CRYPTO 1992. LNCS, vol. 740. Springer, Heidelberg (1993)
4. Cachin, C., Kursawe, K., Shoup, V.: Random oracles in constantipole: practical asynchronous byzantine agreement using cryptography (extended abstract). In: Neiger, G. (ed.) PODC, pp. 123–132. ACM, New York (2000)
5. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin [11], pp. 56–72
6. Chaum, D.: Untraceable electronic mail, return addresses and digital pseudonyms. Communications of the ACM 24(2), 84–88 (1981)
7. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell [3], pp. 89–105

8. Clarkson, M.R., Chong, S., Myers, A.C.: Civitas: A secure remote voting system. Technical Report TR2007-2081, Cornell University (May 2007)
9. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 103–118. Springer, Heidelberg (1997)
10. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
11. Franklin, M. (ed.): CRYPTO 2004. LNCS, vol. 3152. Springer, Heidelberg (2004)
12. El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)
13. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. In: Stern [25], pp. 295–310
14. Hirt, M., Sako, K.: Efficient receipt-free voting based on homomorphic encryption. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 539–556. Springer, Heidelberg (2000)
15. Jakobsson, M., Juels, A.: Mix and match: Secure function evaluation via ciphertexts. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 162–177. Springer, Heidelberg (2000)
16. Jakobsson, M., Sako, K., Impagliazzo, R.: Designated verifier proofs and their applications. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 143–154. Springer, Heidelberg (1996)
17. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: Atluri, V., De Capitani di Vimercati, S., Dingledine, R. (eds.) WPES, pp. 61–70. ACM, New York (2005)
18. Lim, C.H., Lee, P.J.: A key recovery attack on discrete log-based schemes using a prime order subgroup. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 249–263. Springer, Heidelberg (1997)
19. Andrew Neff, C.: A verifiable secret shuffle and its application to e-voting. In: ACM Conference on Computer and Communications Security, pp. 116–125 (2001)
20. Okamoto, T.: Provably secure and practical identification schemes and corresponding signature schemes. In: Brickell [3], pp. 31–53
21. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern [25], pp. 223–238
22. Park, C., Itoh, K., Kurosawa, K.: Efficient anonymous channel and all/Nothing election scheme. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 248–259. Springer, Heidelberg (1994)
23. Schnorr, C.-P.: Efficient signature generation by smart cards. *J. Cryptology* 4(3), 161–174 (1991)
24. Smith, W.D.: New cryptographic voting scheme with best-known theoretical properties. In: Workshop on Frontiers in Electronic Elections (FEE 2005), Milan, Italy (September 2005)
25. Stern, J. (ed.): EUROCRYPT 1999. LNCS, vol. 1592. Springer, Heidelberg (1999)
26. Weber, S.G., Araújo, R., Buchmann, J.: On coercion-resistant electronic elections with linear work. In: 2nd Workshop on Dependability and Security in e-Government (DeSeGov 2007) at 2nd Int. Conference on Availability, Reliability and Security (ARES 2007), pp. 908–916. IEEE Computer Society, Los Alamitos (2007)

Scratch, Click & Vote: E2E Voting over the Internet

Mirosław Kutylowski* and Filip Zagórski**

Institute of Mathematics and Computer Science,
Wrocław University of Technology
mirekk@im.pwr.wroc.pl, filipz@im.pwr.wroc.pl

Abstract. We present *Scratch, Click & Vote* remote voting scheme. The scheme is end-to-end verifiable and allows for voting over the Internet. It guarantees security against malicious hardware and software used by a voter; a voter's computer does not get any knowledge about the voter's choice. Moreover, it can blindly change the voter's ballot with a small probability only.

Keywords: Internet voting, e-voting, E2E, verifiable voting scheme, ThreeBallot, Punchscan.

1 Introduction

There are two main scenarios of e-voting: advanced voting procedures at polling places and remote electronic voting.

Polling station voting. Recently it has become evident that badly designed e-voting machines can be extremely dangerous to a voting process [19,13]. Fortunately, a number of end-to-end auditable voting systems (*E2E*) has been presented recently. Interestingly, some recent designs implement *electronic voting* without any electronic voting machines [2,6,5,4]. Moreover, for these schemes each voter gets a receipt, which may be used to check if the voter's ballot has been included in the tally. It is also possible to verify correctness of the results. On the other hand, the receipt cannot be used even by the voter to prove how she voted. So they cannot help to sell or buy votes.

Internet voting. The Internet voting has not much in common with polling station scenario. At the polling station it is relatively easy to preserve voter's privacy; coercion and vote buying is hard to hide. Another source of problems is that a voter must use some electronic device. There is no convincing argument why a voter should blindly trust this device. Malware can endanger integrity of the elections as well as privacy of the voter.

In case of remote voting one has also to deal with remote voter identification. Fortunately, it can be solved in many ways, depending on a situation. For national elections one can use advanced electronic signatures, especially if supported by personal, government-issued ID cards, or a novel technique described in [3]. For other elections logins and passwords seem to serve well their purpose.

* Partially supported by Polish Ministry of Science and Higher Education, grant N206 2701 33.

** Partially supported by Foundation of International Education (FEM), programme Lower Silesia Research Grants, project *Electronic Identification*.

In this paper we are concerned with an *E2E* systems for remote voting over electronic networks. We assume that the electronic devices used by a voter might be infected by malicious code, and that voter’s privacy and election integrity must be guaranteed in a verifiable way.

1.1 Related Work

Three important ideas concerning *E2E* voting systems have been presented during the last few years: Prêt à Voter, Punchscan, ThreeBallot (and related schemes). All of them are dedicated to paper-based elections at polling stations. Recently, Punchscan and Prêt à Voter have been adjusted to mail-in voting [15]. Since these methods are closely related to our scheme, we recall them briefly.

Prêt à Voter [6]. A voter, say Alice, obtains a ballot which consists of two parts. The left part contains the official list of the candidates, altered by applying a circular shift by x positions, where x depends on the ballot. The right part contains boxes where Alice can put the \times -mark. In order to vote, she puts the \times -mark in the row that contains the name of her favorite candidate on the left side. On the right side there is a kind of ballot serial number S that is used for decoding Alice’s vote (namely, for reconstructing the shift value x). The serial number is also included in the voting receipt obtained by Alice. After making her choice, Alice separates both parts. The left part goes to a shredder, while the right part is scanned and entered to the system.

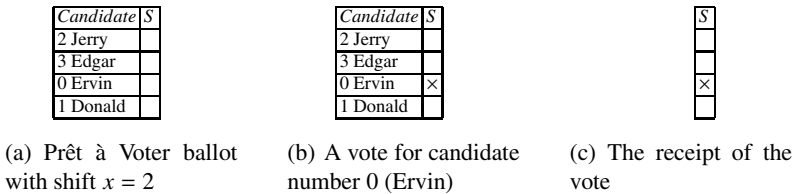


Fig. 1. Ballot example for Prêt à Voter scheme

Punchscan [2]. The original ballot design of Punchscan is quite different from Prêt à Voter, however it has been shown in [19] that the crucial mechanisms of Punchscan can be used together with Prêt à Voter ballot layout.

The key issue is that Punchscan offers a complete back-end to perform *E2E* verifiable elections. Similar back-end is also used in Scantegrity [5] and Scantegrity II [4]. The values that are used in the ballot construction are committed and can be verified. The verification process is twofold and consists of a pre-election audit and a post-election audit. If the authority responsible for preparing ballots passes both audits, then with an overwhelming probability the integrity of the elections is guaranteed.

ThreeBallot [16]. This scheme, presented by R. Rivest, is particularly appealing despite of certain privacy weaknesses [7]. A voter, Alice, obtains a sheet of paper consisting of four parts. The leftmost column contains the list of candidates (no shift is

used). The next three columns are used to mark her choice. If she wants to vote for a candidate V , then she puts two marks \times in the row containing the name of V , while she puts exactly one \times mark in all remaining rows. After Alice makes her choice, all three

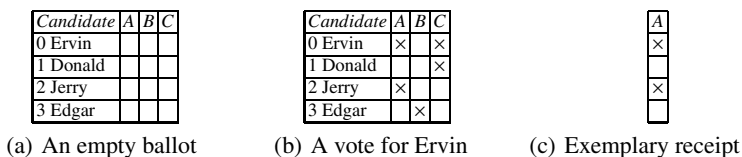


Fig. 2. The ThreeBallot scheme

columns (ballots) are separated and cast into a ballot box. As a receipt Alice obtains a copy of one of the columns/ballots of her choice (but the system does not know which one).

Internet voting schemes. So far, the schemes designed by the academic community do not fulfill all security demands. The most acute problem is that almost all schemes ignore the fact that electronic voting equipment should be considered as a potential adversary ([12],[14]). Meanwhile, potentially the most dangerous in remote voting systems is the equipment on the voter's side. Voters' machines can be infected with malware that reveal the voter's preferences or even change the encrypted ballot cast by the voter.

1.2 Our Contribution

Design goals. The main goal behind design of *SC&V* is to get an *E2E* scheme that would be acceptable for casting votes over Internet. Unlike in the previous works, we demand all key requirements to be satisfied. In particular, the scheme must be secure without assuming that the voter's PC or any of its components is trustworthy. We present the scheme which is:

Human verifiable: A receipt obtained by a voter is human-readable and easy to examine by a moderately educated voter,

Voter friendly: A voter needs not to perform any complicated (and hard to understand by an average voter) operations like: re-encryption, getting a blind signature, executing oblivious transfer protocol etc.

Malware immune: Integrity of the elections and privacy of votes do not rely on any assumption on trustworthiness of the equipment used by the voter,

Efficient: Computational overhead as well as communication volume are low.

Scheme design. In the next sections we provide a full description of *SC&V*. The scheme can be described as a layered design in which we combine a number of techniques/tricks:

Version 1: We start from a straightforward Internet-version of the ThreeBallot: a voter needs to fill in a voting card using an application run on her PC. The filled ballot is then sent to a voting server (*Proxy*).

Version 2: In order to balance the number of \times marks in the columns (in order to make the scheme immune against Strauss’-like attacks [7][10][17][18]) we add another column and another \times -mark (“FourBallot”).

Version 3: Since the voter’s PC knows exactly the voter’s choice – we introduce a coding card which is prepared by a *Proxy* (see the diagram below). The coding card hides from the PC the meaning of the voter’s choices (every candidate is “clicked” exactly once). Moreover coding card is constructed in a way that the possibility of modification of voter’s choice by her PC is reduced.

Version 4: Since *Proxy* still knows the voter’s choice, we introduce another server – Election Authority (EA), which is responsible for preparation of ballots. In this way *Proxy* does not know the choice, while Election Authority does not know who voted.

Election Authority prepares voting cards in a similar way as for the Punchscan scheme. Together with the ThreeBallot mechanisms this ensures verifiability of the voting process.

Under the assumption that *Proxy* and *Election Authority* do not collude, *SC&V* offers verifiable Internet voting with unconditional integrity and full privacy of a voter.

2 Ideas Overview

2.1 Ballots and Coding Cards

The voting process is a combination of paper based protocols. In order to vote one has to get additional information that remains hidden from the computer used for vote casting. This information might be obtained by the voter during voter’s in-person registration, mailed to her home address or sent over an independent electronic link. In case of small scale elections or elections of limited importance one can use emails with CAPTCHA to sent information readable for a human voter, but hard to read by the voter’s computer. Vote casting is done via electronic networks.

There is an *Election Authority (EA)* and a *Proxy*. *EA* prepares the *ballots* while *Proxy* prepares *coding cards*. There is an *Auditor* which is responsible for pre- and post-election audits.

In this section we describe the scheme from the point of view of a voter Alice. For the sake of simplicity of exposition we assume that there is a single race where the voter has to choose one out of m candidates (the pictures presented below depict the case of $m = 4$).

Ballot layout. In order to cast a vote, Alice needs a *ballot* and a *coding card*.

The **ballot** is prepared by *EA*, it consists of the following values covered by a scratch surface:

- list of candidates permuted with some random permutation π . Later we shall represent $\pi = \pi' \circ \pi''$ where π', π'' are random.
- ballot serial number S_I ,
- four *confirmation tokens*: A, B, C, D – one per column. They are prepared in a special way that will be described below.

Candidate	A	B	C	D
2 Jerry				
3 Edgar				
0 Ervin				
1 Donald				
S_I				

Fig. 3. A ballot with $\pi(i) = i + 2$

The **coding card** is prepared by *Proxy* and consists of:

- four columns. In each row there is exactly one mark Y standing for YES, and 3 marks n standing for NO. The placement of Y in each row is random and independent from the other choices.
- coding card serial number S_r .

	n	Y	n	n
	n	Y	n	n
	Y	n	n	n
	n	n	n	Y
S_r				

Fig. 4. A coding card

2.2 Voter's Point of View

Alice obtains both: the ballot and the coding card¹. Alice lays them side by side and thus obtains a complete ballot. Let us note that Alice gets exactly one ballot, but she is allowed to have as many coding cards as she likes. Moreover, we assume that there are many Proxies in the system², so Alice can easily find one she trusts and gets coding cards from this Proxy. A complete ballot (which Alice may put on her desk) may look as follows:

complete ballot	PC screen	ballot matrix	receipt
<i>Candidate</i> A B C D			
2 Jerry n Y n n	■ □ □ □ □	□ □ × ×	×
3 Edgar n Y n n	□ ■ □ □ □	× □ □ ×	□
0 Ervin Y n n n	■ □ □ □ □	□ × × ×	×
1 Donald n n n Y	□ ■ □ □ □	× □ × □	×
S_l S_r	S_r	S_l	C, c

Alice visits an election website operated by the *Proxy*. She authenticates herself with appropriate authentication method (login/password pair, electronic signature etc). She clicks on the screen in the following way:

- she clicks on the position of Y in the row corresponding to the candidate that she votes for,
- in each of the remaining rows, she clicks on one of the positions of n's.

The *Proxy* commits to Alice's clicks (the commitment is passed to *EA*), then Alice enters coding card serial number S_r . The *Proxy* checks S_r and then transforms the choice of the voter into an internal form called *ballot matrix*: *Proxy* puts × mark for each n which has not been used yet (this transformation depends deterministically on the positions of Y's and n's and the voter's choice). So, for a row with the candidate chosen by the voter *Proxy* puts three × marks, while in each row corresponding to different candidates, there are only 2 × marks. Note that *Proxy* knows which row corresponds to the vote cast. On the other hand, due to the random permutation *Proxy* does not know which candidate is corresponding to this row.

Then the columns of the ballot matrix, called *ballot columns*, are processed separately (analogously to ThreeBallot). In the next step *Proxy* obtains a blind signature (BS) of *EA* under each ballot column. (A blind signature is necessary in order to prevent

¹ Alice obtains it at registration office, by mail or by email, depending on election settings.

² Moreover, a “decoy service” can be introduced – then Alice may obtain many different but fake coding cards with the same serial number – in order to cheat a coercer or a vote-buyer.

changing the ballot contents by EA at this moment). The voter enters ballot serial number (S_l), then $Proxy$ unblinds the signature, and sends ballot columns with S_l to EA . Simultaneously, the voter requests one ballot column as a receipt. The receipt contains:

- $T \in \{A, B, C, D\}$ value of a confirmation token;
- y - ballot column,
- t such that $T = \text{sign}_{EA}(t, S_l)$, such t is called a pre-token of T .

The ballot columns are now separated and published just like for Punchscan scheme, and then decrypted in a similar way. The number of votes for each candidate is counted like for ThreeBallot.

3 Scratch, Click & Vote Scheme

The Ballots and Audit Tables. The ballots are created by EA . In order to guarantee election integrity, EA generates audit tables P and R (see below for details). Each row of table P corresponds to a single ballot matrix (which is a set of columns with the same ballot serial number and the permutation of the candidates). Entries of R correspond to single ballot columns. R is based on the same idea as Punchboard used in Punchscan.

Table P. The table P has 2 columns, called P1 and P2. It has $2n$ rows, where n is greater or equal to the maximum number of voters.

P1	P2
⋮	⋮
⋮	⋮
$S_l(i)$	$BC(i_A), BC(i_B), BC(i_C), BC(i_D)$
⋮	⋮
⋮	⋮

Example: Audit table P

The column P1 records the ballot serial numbers. The column P2 contains commitments to 4 pointers to the rows of table R . Say, if a serial number S is in P1, then in the same row, column P2 contains commitments

$$BC(i_A(S)), BC(i_B(S)), BC(i_C(S)), BC(i_D(S))$$

to numbers $i_A(S), i_B(S), i_C(S), i_D(S)$, where $i_X(S)$ is the row number such that row $i_X(S)$ of table R contains an entry for the column X of the ballot with the serial number S .

Table R. The table R consists of three parts: the *starting part*, the *middle part* and the *final part*. Each part consists of a set of consecutive columns. R has $8n$ rows; this corresponds to $2n$ ballots and $4 \cdot 2n$ ballot columns. There are two types of permutations used for constructing table R :

- ρ_1, ρ_2 : permutations of rows of the R (i.e. permutations over $\{1, \dots, 8n\}$),
- permutations π'_i, π''_i for $i = 1, \dots, 8n$ over $\{1, \dots, m\}$, where m is the number of candidates (i.e. 2 permutations per each of $8n$ ballot columns). These permutations have to be applied to ballot columns.

Each row i in the starting part of R is devoted to a single ballot column of some ballot, (and for each ballot column from some ballot there is exactly one such a row of R). Let $W(i)$ denote the ballot column corresponding to the i th row of R . Then for each i , data concerning $W(i)$ are placed in:

- row i of the starting part,
- row $\rho_1(i)$ of the middle part,
- row $\rho_2(\rho_1(i))$ of the final part.

Moreover:

- the starting part of row i will contain the ballot column $W(i)$ as filled by the voter (the order of the candidates is determined by $\pi_i = \pi'_i \circ \pi''_i$, i.e. the entry for a candidate j is given in row $\pi_i(j)$,
- the middle part at row $\rho_1(i)$ will contain $W(i)$ permuted by $(\pi'_i)^{-1}$,
- the final part at row $\rho_2(\rho_1(i))$ will contain $W(i)$ permuted by $(\pi''_i)^{-1} \circ (\pi'_i)^{-1}$. Hence the marks of $W(i)$ will be permuted according to the standard ordering of candidates: $(\pi''_i)^{-1} \circ (\pi'_i)^{-1} \circ \pi_i = (\pi''_i)^{-1} \circ (\pi'_i)^{-1} \circ \pi'_i \circ \pi''_i = id$.

Below we describe the i th row of R . Let $i = \rho_1(j)$ and $i = \rho_2(\rho_1(k))$.

starting part (for $W(i)$)					middle part (for $W(j)$)				final part (for $W(k))$	
i	$\widehat{\pi}_i$	$H(t(i))$	$\widehat{t(i)}$	$y(i)$	$\widehat{\rho_1(i)}$	$\widehat{\pi'_j}$	$y(j) \ll (\pi'_j)^{-1}$	$\widehat{\pi''_j}$	$\widehat{\rho_2(i)}$	v

Organization of a row of table R

The starting part contains the following entries in row i (see the diagram above):

- i – the row index ($i \in [1, 8n]$),
- $\widehat{\pi}_i$ – a bit commitment to permutation of candidates π_i used in the ballot containing $W(i)$,
- $H(t(i))$ – hash of a *confirmation pre-token* $t(i)$, which satisfies the condition

$$T(i) = \text{sign}_{EA}(t(i), S_l(i)),$$

where $T(i)$ is the confirmation token used in conjunction with $W(i)$, and $S_l(i)$ is the serial number of the ballot containing $W(i)$,

- $\widehat{t(i)} = BC(T(i), S_l(i))$ – a bit commitment to the ballot serial number $S_l(i)$ of the ballot containing $W(i)$, and to the confirmation token $T(i)$,
- $y(i) = [y_0(i), y_1(i), \dots, y_{m-1}(i)]$ – a vector holding mark \times on those positions l such that $W(i)$ contains the \times mark in row l . Initially, during creation of table R , the vector $y(i)$ is empty. It becomes filled after casting a vote.
- $\widehat{\rho_1(i)}$ – a commitment to the value $\rho_1(i)$.

The middle part of R in row i contains the following entries:

- $\widehat{\pi'_j}$ – a commitment to permutation of candidates π'_j , where $\pi_j = \pi'_j \circ \pi''_j$,
- $y(j) \ll (\pi'_j)^{-1}$ the vector $y(j)$ permuted by $(\pi'_j)^{-1}$,
- $\widehat{\pi''_j}$ – a commitment to permutation π''_j ,
- $\widehat{\rho_2(i)}$ – a commitment to $\rho_2(i)$.

The final part of R in row i contains vector v equal to $y(k)$ permuted by $(\pi''_k)^{-1}$ and then by $(\pi'_k)^{-1}$ (i.e., listed according to the standard ordering of the candidates).

Preparation of Ballots and Audit Tables. The ballots and the audit tables P and R are created by EA in the following way:

1. EA determines the election parameters: the number of candidates m , the official list of candidates (with their official ordering), and an upper bound n on the total number of voters.
2. EA chooses at random $2n$ serial numbers; for each serial number S :
 - EA chooses at random a random permutation π ,
 - EA chooses at random confirmation pre-tokens $t_A(S)$, $t_B(S)$, $t_C(S)$, $t_D(S)$ and computes confirmation tokens $T_A(S)$, $T_B(S)$, $T_C(S)$, $T_D(S)$

$$T_X(S) := \text{sign}_{EA}(S, t_X(S)) \text{ for } X = A, B, C, D .$$

3. EA creates audit table P : For this purpose, EA chooses at random a permutation σ of $1, \dots, 8n$. Then $\sigma(4j - 3), \dots, \sigma(4j)$ are assigned to the j th serial number $S_l(j)$. These numbers serve as pointers to the rows of the audit table R - and are called $i_A(S_l(j))$, $i_B(S_l(j))$, $i_C(S_l(j))$, $i_D(S_l(j))$. Then for each serial number $S_l(j)$, commitments to the values $i_A(S_l(j))$, $i_B(S_l(j))$, $i_C(S_l(j))$, $i_D(S_l(j))$ are created and inserted in the row containing $S_l(j)$.
4. EA prepares the audit table R : For this purpose EA chooses random permutations (on R -table rows) ρ_1 and ρ_2 of $1, \dots, 8n$. For the j th serial number $S_l(j)$, its permutation π (on ballot columns) is assigned to the rows $i_A(S_l(j))$, $i_B(S_l(j))$, $i_C(S_l(j))$, $i_D(S_l(j))$ of the starting part of R . (i.e., $\pi_{i_A(S_l(j))}$, $\pi_{i_B(S_l(j))}$, $\pi_{i_C(S_l(j))}$, and $\pi_{i_D(S_l(j))}$) take the value π). Separately for each row i of R , EA chooses at random permutations π'_i and π''_i such that $\pi_i = \pi'_i \circ \pi''_i$.
5. Then the entries of R are filled according to the description from the previous subsection.

Finally, the ballots are printed so that their contents (the permutation of the list of candidates names, confirmation tokens and serial numbers) is hidden under a scratch layer.

The Pre-election Audit. As for Punchscan, the following steps are executed in order to check that the audit tables have been created honestly:

1. The Auditors pick at random a set AS of n ballots. The remaining ballots create a so called election set ES (and are not checked).
2. The contents of all ballots from AS is revealed, so in particular their serial numbers. Based on the serial numbers it is possible to indicate the rows of P corresponding to the ballots from AS .
3. EA opens all bit commitments from table P corresponding to the ballots from AS as well as all bit commitments from table R corresponding to the ballot columns of the ballots from AS .
4. The Auditors check whether the ballots and the entries in the audit tables were created correctly.
5. All ballots from the audit set AS are discarded; the ballots with serial numbers in ES are used for election.

In practice, the Auditors may confine themselves to controlling only a limited number of ballots from AS , and check more ballots on demand.

Preparing Coding Cards. The coding cards are prepared in an electronic form and are published (as commitments) on a webpage by the *Proxy*. Their correctness is checked in a standard way:

1. *Proxy* creates an audit table X in which it commits to coding card serial numbers S_r and positions of Y-marks on each coding card.
2. The Auditors select at random some number of coding cards to form an audit set (these coding cards are not used for elections).
3. *Proxy* opens all bit commitments from the cards of the audit set.
4. The Auditors check if the revealed coding cards have been created correctly.

Elections. The following steps are during vote casting:

Step 1: The voter obtains a ballot (e.g. by visiting certain authorities, from a special courier delivering the ballots at residence area, by certified mail services etc.). At the same time, identity of the voter is verified and the ballot is given to her own hands. Distribution of ballots is organized so that nobody knows who gets which ballot. Since the ballot information is covered with a scratch surface, this is easy.

Step 2: The voter fetches (still unused) coding cards from one or more Proxies (for convenience the coding cards can be printed).

Step 3: The voter peels-off the scratch-layer from the ballot.

Step 4: The voter logs in an election webpage run by a *Proxy* and authenticates herself.

Step 5: Proxy verifies voter's credentials.

Step 6: The voter chooses one of the coding cards and lays it next to the ballot.

Step 7: The voter clicks on the PC screen on radio buttons corresponding to her choice – that is, according to the permutation used for the ballot and alignment of n and Y 's marks on the coding card.

Step 8: Proxy commits to voter's clicks, sends the commitment to EA and to the voter (so the voter can print it³).

Step 9: The voter enters S_r from the coding card used.

Step 10: Proxy transforms the voter's choice into ballot columns.

Step 11: Proxy obtains a blind signature from EA under each of the ballot columns (these signatures are then stored by *Proxy* for a post-election audit).

Step 12: The voter enters S_l .

Step 13: Proxy passes S_l and the ballot columns to EA .

Step 14: EA enters the obtained ballot columns into appropriate rows of the starting part of table R (but EA publishes them when the election are closed), EA publishes commitments to the ballot columns obtained from *Proxy*.

Step 15: The voter chooses a receipt (one of the four columns).

Tallying

1. When the voting time is over, EA publishes voter's choices inserted into vectors $y(i)$ in the starting part of the table R . Then it computes the entries for the middle part of R : $y(j) \ll (\pi'_j)^{-1}$, and for the final part:

$$v = (y(k) \ll (\pi'_k)^{-1}) \ll (\pi''_k)^{-1}.$$

³ This commitment can be later used during investigation in the case if a fraud was detected.

2. From the entries v in the final part EA calculates the tally: If the number of ballot columns is $4N$ (meaning that N votes have been cast) and there are together M marks \times in row j of all ballot columns in the final part of R , then the number of votes cast for the j th candidate is $M - 2N$.

Post-Election Audit. First, each voter can check if her ballot column corresponding to the receipt appears in the table R . This is possible, since due to knowledge of the verification pre-token t , one can locate the right row containing $H(t)$. If it is missing or the contents of the ballot column disagrees with the receipt, then a fraud is detected.

Checking integrity of table R and the election results is performed in public by the auditors. For this purpose the standard Randomized Partial Checking [11] procedure is executed for R (for the sake of simplicity of description we assume that n voters participated in the elections):

1. The auditors choose $2n$ rows of R at random and request EA to open commitments $\widehat{\rho}_1(i)$ from these rows. Then for each row $\rho_1(i)$ in the middle part, for which $\rho_1(i)$ has been revealed, the commitment $\widehat{\pi}_i'$ is opened and it is checked that the ballot column from the starting part permuted by $(\pi_i')^{-1}$ yields the ballot column in the middle part.
2. For each row j in the middle part, not pointed to by any revealed commitment $\rho_1(i)$, EA has to open the commitments to $\rho_2(j)$ and $\widehat{\pi}''$. Then the ballot columns in the middle part of row j permuted with π''^{-1} and the ballot column in the final part of row $\rho_2(j)$ should be equal.

4 Security Concerns

Voter's PC misbehaviour. Here we assume that the Alice's PC is dishonest, while EA and *Proxy* behave correctly. This corresponds to the case when Alice's PC is infected by malware.

Integrity. In order to manipulate voter's choice (change from Alice's choice to any other candidate, even a random one) the PC has to switch Alice's choice from Y into n in the row corresponding to the candidate chosen by Alice and at the same time, change n into Y in one of the remaining rows. In order to do that, the PC has to guess which row corresponds to the chosen candidate and succeeds with probability $\frac{1}{k}$. Then, the PC has to choose one of the remaining rows and guess which one of the unchosen three columns corresponds to the mark Y – this succeeds with probability $\frac{1}{3}$.

So the probability of correct switching Alice's choice is only $\frac{1}{3k}$. But even then, Alice can still detect a fraud by discovering that her receipt does not fit her choice. At least one of the ballot columns is modified during such change (and sometimes it is just one column). So the total probability of a successful and undetected vote change is $\frac{1}{4k}$.

Privacy. Even if Alice's PC sends all information it is aware of to an attacker he is unable to determine the choice of Alice. Indeed, the attacker neither knows the configuration of Y 's on the coding card nor the permutation used on the ballot.

EA misbehaviour. Now assume that *EA* is dishonest, but the PC of Alice and the *Proxy* are honest.

Integrity. Misbehaviour in ballots' preparation and counting is limited by the pre- and post-election audits just like in the case of Punchscan.

Replacing ballot columns when inserting them to table *R* is risky, since the voter gets a receipt, which is one of her four ballot columns signed by *EA*. If the receipt disagrees with the contents of table *R*, then one can catch *EA*. Recall that the ballot columns are signed blindly by *EA* before *EA* knows ballot's serial number (and thus the permutation) and that *EA* does not know which of the ballot columns is chosen for the receipt. Note that since the hash value of the pre-token is posted in *R*, the voter can prove which entry in the starting part of *R* corresponds to the ballot column from her receipt.

Privacy. The situation is like for Punchscan: If *EA* knows which ballot was used by Alice, then it knows the vote cast by Alice. So it is crucial to apply appropriate procedures of ballot distribution. Keeping the sensitive information under scratch surface is a good solution - the ballots can be mixed before distribution, becoming thus indistinguishable. Also, it is crucial that voters never send ballot information directly to *EA* - all communication must go through *Proxy*.

Proxy's misbehaviour. Now we assume that *Proxy* is dishonest, while *EA* and the PC of Alice behave correctly.

Integrity. *Proxy* commits to the voter's clicks before it knows S_r , so *Proxy* cannot change voter's choice.

If *Proxy* changes S_l in order to change the permutation of a ballot then Alice obtains a confirmation token that is different from the one stated on her ballot. Thus it will be detected immediately by Alice.

Privacy. The assignments of *Y*'s and *n*'s are known to *Proxy*, so *Proxy* knows the row corresponding to the voter's choice. However, *Proxy* does not know the permutation used in the ballot, so it cannot link the vote with any particular candidate.

External observer's point of view. Here, we assume that the observer Charlie is not physically present during casting a vote and does not control the PC used by Alice. We assume that Alice casts the vote and then passes to Charlie (e.g. by mail or fax): the ballot, the coding card and the receipt and informs which fields have been clicked. Of course, Charlie has access to the bulletin board.

The key point is that Alice could use a coding card different from the one she shows to the observer - receipt does not contain S_r . So, the situation of Charlie is much different from the situation of a *Proxy*: Charlie obtains only one ballot column (receipt) and cannot be sure if the coding card obtained was really used.

Vote selling. In case of *SC&V* voter is identified electronically by the *Proxy*. The identification protocol should guarantee that the voter would not risk transmitting her electronic identity to the buyer. (In this way *SC&V* becomes superior over postal procedures, for which transferring a ballot to a buyer cannot be prevented.) This holds for

instance, if the voter is using an electronic ID card or ID codes that are used also for other purposes (like submitting a tax declaration).

Even if Alice casts the vote herself, she can record the whole voting session and present it to the buyer together with the ballot and the coding card used. The ballots have a non-electronic, paper form, so they can be presented to the remote buyer as electronic copies. However, the scan of the ballot can be manipulated and the coding card presented needs not to be the one actually used.

Things become more complicated for the buyer, if the authentication protocol is based on a zero-knowledge protocol – then the buyer cannot be even sure that the voter is casting a vote unless he is controlling directly the voter’s PC.

The only thing that the buyer can be convinced about is the receipt and the matching entries in the bulletin board. However, at this moment we fall back to the case of the external observer considered above.

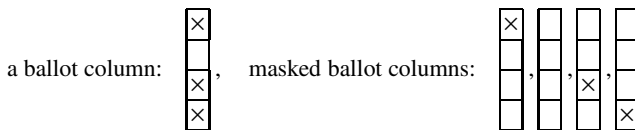
Decoy service. Decoy service can be optionally launched in order to make vote-buying and coercion harder. A voter can visit a webpage of a decoy service and download fake coding card with any given serial number and arbitrarily chosen arrangement of Y’s and n’s. Then, the voter may present such a coding card to a coercer or a vote buyer together with a receipt and ballot to prove that she voted in a certain way.

Usability issues. The scheme might be uneasy to use at least for two reasons. First, a voter needs to click next to every candidate. Second, a voter needs to find her candidate on a permuted list of candidates. So use of shifted lists of candidates (as for Prêt à Voter) can improve usability. Unfortunately, shifts have bad impact of privacy.

A receipt obtained by a voter is an k -row column. If permutations are used for re-ordering lists of candidates, then there are $k + 1$ types of receipts (any two receipts with the same number of \times marks are equivalent).

The situation is different, if shifts are used. Then, the number of different types of receipts is equal to the number of k -bead necklaces with 2 colors which equals to (see [8]): $S(k) = \frac{1}{k} \sum_{d|k} \varphi(d) 2^{\frac{k}{d}}$, where $\varphi(k)$ is the Euler quotient function, i.e. for k -prime $S(k) = 2^k/k$. In this case, in order to achieve privacy of votes, the number of voters has to be much higher than $S(k)$. The other solution to this problem is to use so called *masked ballot columns*.

The idea is that table R stores in its rows instead of *ballot columns* – k corresponding *masked ballot columns*. Simply, the j th *masked ballot column* of a given *ballot column* contains no \times mark except for the row j , provided that the original ballot column contains a \times mark in the row j . See an example of a ballot column and its masked versions:



Let X be a ballot column and t be its pre-token. Then the j th masked ballot column for X in the starting part of R is marked by the value $H(t, j)$ (instead of $H(t)$, as it was for the first design). This enables the voter to check the entries of the bulletin board as before - however the voter has to look for k different hashes and k rows instead of one.

Checking integrity of R table is performed just as before, as well as vote counting: the number of \times marks does not change, the number of votes is now the number of rows of table R divided by $4k$.

How do the masked ballot columns help to preserve anonymity? The key observation is that the number of masked ballot columns of each kind is determined **uniquely** by the election result. Therefore R table provides **no** additional information. So the Strauss' attack and any other attack based on the particular choice of ballot columns fails.

The same technique may be applied to ThreeBallot scheme.

5 Final Remarks

SC&V allows for secure and verifiable vote casting over the Internet with unconditional integrity. Privacy is preserved with the assumption that both authorities do not collude.

A voter cannot prove how she voted unless vote-casting is physically supervised by an adversary (it is not the case in Internet version of Punchscan [15]).

Online vote-selling is almost impossible. In order to buy a vote, a buyer needs to obtain:

- the record of a voting session from the voter's computer (the serial numbers of ballot and coding card, and the voter's choices),
- ballot and coding card used.

Even if the voter's PC is infected by viruses, her choice remains secret. Moreover, any attempt of modification of voter's choice is detected with high probability.

References

1. Top-to-bottom review. top-to-bottom report conducted by Secretary of State Debra Bowen of many of the voting systems certified for use in California (2007), http://www.sos.ca.gov/elections/elections_vsr.htm
2. Chaum, D.: Punchscan (2005), <http://www.punchscan.org>
3. Chaum, D., Clarkson, M.R., Haber, S., Jakobsson, M., Popoveniuc, S., Zagórski, F.: Internet voting as secure as polling-place voting (preprint)
4. Chaum, D., Essex, A., Carback, R., Clark, J., Popoveniuc, S., Rivest, R.L., Ryan, P.Y.A., Shen, E., Sherman, A.: Scantegrity ii: End-to-end voter-verifiable optical scan election systems using invisible ink confirmation codes. In: USENIX/ACCURATE EVT 2008 (2008)
5. Chaum, D., Essex, A., Carback, R., Clark, J., Popoveniuc, S., Sherman, A., Vora, P.: Scantegrity: End-to-end voter-verifiable optical- scan voting. IEEE Security and Privacy 6(3), 40–46 (2008)
6. Chaum, D., Ryan, P.Y.A., Schneider, S.: A practical voter-verifiable election scheme. In: di Vimercati, S.d.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 118–139. Springer, Heidelberg (2005)
7. Cichoń, J., Kutyłowski, M., Węglorz, B.: Short ballot assumption and threeballot voting protocol. In: Geffert, V., Karhumäki, J., Bertoni, A., Preneel, B., Návrat, P., Bieliková, M. (eds.) SOFSEM 2008. LNCS, vol. 4910, pp. 585–598. Springer, Heidelberg (2008)
8. Flajolet, P., Sedgewick, R.: Analytic combinatorics (2008)

9. Gogolewski, M., Klonowski, M., Kutylowski, M., Kubiak, P., Lauks, A., Zagórski, F.: Kleptographic attacks on E-voting schemes. In: Müller, G. (ed.) ETRICS 2006. LNCS, vol. 3995, pp. 494–508. Springer, Heidelberg (2006)
10. Henry, K., Stinson, D.R., Sui, J.: The effectiveness of receipt-based attacks on threeballot. Cryptology ePrint Archive, Report 2007/287 (2007), <http://eprint.iacr.org/>
11. Jakobsson, M., Juels, A., Rivest, R.L.: Making mix nets robust for electronic voting by randomized partial checking. In: USENIX Security Symposium, pp. 339–353 (2002)
12. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: WPES 2005, pp. 61–70 (2005)
13. Karlof, C., Sastry, N., Wagner, D.: Cryptographic voting protocols: A systems perspective. In: USENIX Security Symposium, pp. 33–50 (2005)
14. Moran, T., Naor, M.: Receipt-free universally-verifiable voting with everlasting privacy. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 373–392. Springer, Heidelberg (2006)
15. Popoveniuc, S., Lundin, D.: A simple technique for safely using punchscan and prêt à voter in mail-in elections. In: Alkassar, A., Volkamer, M. (eds.) VOTE-ID 2007. LNCS, vol. 4896, pp. 150–155. Springer, Heidelberg (2007)
16. Rivest, R.L., Smith, W.D.: Three voting protocols: Threeballot, vav, and twin. In: EVT 2007: Proceedings of the USENIX/Accurate Electronic Voting Technology on USENIX/Accurate Electronic Voting Technology Workshop, Berkeley, CA, USA, p. 16. USENIX Association (2007)
17. Strauss, C.: A critical review of the triple ballot voting system. part 2: Cracking the triple ballot encryption (2006), <http://www.cs.princeton.edu/appel/voting/Strauss-ThreeBallotCritique2v1.5.pdf>
18. Strauss, C.: The trouble with triples: A critical review of the triple ballot (3ballot) scheme, part 1 (2006), <http://www.cs.princeton.edu/appel/voting/Strauss-TroubleWithTriples.pdf>
19. van de Graaf, J.: Merging pret-a-voter and punchscan. Cryptology ePrint Archive, Report 2007/269 (2007), <http://eprint.iacr.org/>

Securing Optical-Scan Voting

Stefan Popoveniuc¹, Jeremy Clark², Richard Carback³,
Aleks Essex⁴, and David Chaum⁵

¹ George Washington University

² University of Waterloo

³ University of Maryland, Baltimore County

⁴ University of Ottawa

⁵ Voteegrity

Abstract. This paper presents a method for adding end-to-end verifiability to any optical-scan vote counting system. A serial number and set of letters, paired with every candidate, are printed on each optical-scan ballot. The letter printed next to the candidate(s) chosen by the voter is posted to a bulletin board, and these letters are used as input to Punchscan’s verifiable tallying method. The letters do not reveal which candidate was chosen by the voter. The method can be used as an independent verification mechanism that provides assurance that each vote is included in the final tally unmodified—a property not guaranteed by a manual recount. We also provide a proof-of-concept process that allows the election authority to settle disputes after the polls close while preserving ballot secrecy.

Keywords: anonymity, cryptography, E2E, mix networks, optical-scan, privacy, Punchscan, security, universal verifiability, voting.

1 Introduction

Abraham Lincoln once observed that democracy is “the government of the people, by the people and for the people.” The foundation of any government *by the people* rests on a society’s ability to hold inclusive elections and accurately count every vote. Unfortunately, the introduction of new voting technology in some countries, including the United States, has diminished voters’ confidence in the security of their democratic contribution.

At the time of writing, the most prominent voting technology used in US elections are optical-scan systems [EDS06]. These systems provide two methods for counting votes. Under precinct scanning¹, the scanned ballots are electronically tallied at the precinct after the polls close. Tallies produced this way rest on software security, and recent security reviews of certified optical-scan systems have demonstrated serious vulnerabilities that undermine the trustworthiness of tallying through this method [WJB06, KMRS06, ea07]. The second tallying

¹ Alternatively the ballots can be scanned centrally, however this requires greater reliance on chain-of-custody.

method is to conduct a manual recount. Hand counting ballots is slow and prone to human error, but most seriously, manual recounts do not detect if ballots were replaced or modified.

This paper² proposes a simple way to add a third, superior vote counting method to any optical-scan voting system—a method that is *end-to-end* (E2E) verifiable. Our method does not interfere with the mechanics of the optical-scan procedures. It permits voters to mark the ballot exactly as in the underlying system, candidates can be listed in a fixed order, the electronic tally can be performed on the same equipment, and manual recounts can be conducted as necessary. Our method only requires the central election authority to print additional information on each ballot and follow some additional procedures after the polls close. Procedures at the precinct remain essentially unchanged. Thus, we introduce no additional risk of equipment failure and only generate a marginal increase in cost. In return, each voter gains the ability to check her choices are included in the tally unmodified and everyone can check that all included ballots are counted correctly.

Our method evolved from the Punchscan voting system [PH06], which contains two key elements: (1) The front-end which is the ballot, how it is constructed, marked, scanned, and posted; and (2) the back-end which provides a universally verifiable method for recovering voter choices from the posted information without compromising ballot secrecy. We introduce significant modifications to the front-end by pairing symbols with candidate names on each ballot. The symbol paired with each chosen candidate is posted on a bulletin board. The back-end only changes semantically by decoding what each letter means in a verifiable way instead of decoding a position marked on the ballot. In addition, we provide a dispute resolution process which can be conducted after the polls close.

This paper is organized as follows. Preliminaries are provide in the next section. Section 3 outlines our motivation for creating this method and our design goals for the new ballot style which is introduced in Section 4. The voter experience at the precinct is explained in Section 5, while section 6 gives an overview of how the tally is generated using the Punchscan back-end. Section 7 presents a dispute resolution process for proclaimed discrepancies between a voter's receipt and the bulletin board. Finally, Section 8 provides implementation details and Section 9 contains our concluding remarks.

2 Related Work

End-to-End (E2E) voting systems (also known as receipt-based or universally-verifiable voting systems) represent a class of systems that offer unconditional integrity of election results, usually by using cryptographic techniques. They include VoteHere's MarkPledge [Nef04a, Nef04b], Punchscan [ECCP07], Prêt

² This paper is derived from a presentation by the first author at *Frontiers of Electronic Voting* in 2007. Conceptually, the ideas herein predate a more thorough treatise of the system to appear in *IEEE Security and Privacy*.

à Voter [CRS04], Scratch & Vote [AR06] and Voter Initiated Auditing [Ben07]. Typically, an E2E system will provide the voter with a signed or stamped receipt of her vote. To preserve ballot secrecy, the receipt does not reveal how she voted but contains an indirect representation of her vote—either through encryption or a permutation-based obfuscation. The polling place equipment records this same indirect representation and never sees the voter’s actual choices. After the polls close, the election authority publishes all of the representations it received on a public bulletin board, allowing voters to check that their choices are included and unmodified. In the case of an error, the signature on the receipt provides the voter with proof of a discrepancy—an event that can trigger a variety of responses, depending on election policy.

The election officials will generate the final tally by recovering the votes from the indirect representations of the ballots, either through decryption or inverting the obfuscating permutation. To preserve voter privacy, the tally is generated through a specially designed protocol that explicitly removes the correspondence between a receipt and the choices it represents. To ensure the unconditional integrity of this protocol, a mandatory auditing process is performed on the protocol which proves to a mathematical certainty that all ballots were recovered properly and that the choices were unmodified—whether by a software error, a malicious election official, or a hacker. This auditing process can be independently duplicated by anyone.

3 Design Goals

The front-end of Punchscan has been criticized for its use of indirection, which could introduce voter intention errors and significantly increase time-to-vote. Filling out a Punchscan ballot requires the voter to find a letter corresponding to the candidate of choice in a row of randomly ordered symbols and then marking the corresponding hole. To date, no peer-reviewed user study has been performed on the usability of a Punchscan ballot and so these criticisms are unsubstantiated, however they appear reasonable.

A trade-off involved with the use of Punchscan is that it cannot produce on-demand ballots. This is because it requires paper with holes punched in unique positions for each election. While drilling holes through paper is very fast and cheap, it does require special equipment. The system is also environmentally wasteful, requiring the destruction of one of the two paper sheets composing the ballot.

Another issue is that custom software to scan a Punchscan ballots is needed. While current optical-scan voting equipment could be used to acquire an image of the ballot, the hardware was developed to perform mark sense scanning, which detects whether a shape has been filled in or an arrow has been drawn. The software for scanning Punchscan ballots is simple but the cost and inconvenience of upgrading the software on already-owned optical scanners is a serious impediment to the adoption of Punchscan.

Punchscan uses an unfamiliar mechanism for counting votes. While it provides exceptionally high integrity, it does not comply with legislation and voting

standards that require hand-countable paper ballots for manual recounts. A preference for elements that voters and election officials are used to could ultimately weigh the decision to adopt new voting technology away from systems like Punchscan. Having only minimal changes that can be easily explained and understood by those running the elections is critical for the success of the adoption process.

With regards to privacy, Punchscan ballots are better than optical-scan ballots. With Punchscan, the scanner does not get to see the full ballot, only the receipt which does not indicate how the voter voted. Additionally, any fingerprints that are left by the voter on the scanned paper are irrelevant. However this makes it impossible to have results reported by the precinct optical scan or to conduct a manual recount. Even though Punchscan ultimately offers a stronger proof of accuracy than even a manual recount, this is a legal impediment to Punchscan's adoption in many jurisdictions.

Taking these criticisms, trade-offs, and shortcomings into consideration, we have produced a set of design goals for a new ballot style that can interface with Punchscan's back-end. This is not meant as a replacement for Punchscan, nor should it be thought of as an improvement in all regards. Some jurisdictions may opt for the improved privacy properties offered by Punchscan *in lieu* of the ability to conduct manual recounts or use existing equipment. Because the back-end is shared with Punchscan, ballots could be mixed and matched in the same election. Some ballots can be printed in Punchscan style, while others can be of the new type. To summarize, our design goals are as follows:

1. Eliminate indirection,
2. Allow on-demand printing,
3. Use a single sheet,
4. Use a familiar method for marking the ballots,
5. Allow the use of existing voting equipment without upgrades,
6. Do not interfere with optical-scan tallying,
7. Do not preclude the option of a manual recount,
8. Allow Punchscan's privacy-enhanced ballots to be used in conjunction.

4 Ballot Design

As will become evident, one major advantage of our technique is the ability to preserve the ballot layout that is imposed by the law or an equipment manufacturer. With our method, the order in which the candidates appear on the ballot can be the same on each ballot. We illustrate a typical optical scan ballot configuration in Figure 1. Our method adds two elements to this ballot: symbols that are paired with each candidate and a serial number represented in a form that a mark sense scanner can read.

Before the election, a set of symbols is published for every contest on the ballot. These symbols can be letters, numbers, shapes, or multi-character codes. A fixed *canonical order* of the symbols, corresponding to candidate order, is also established. The order itself is arbitrary and we use alphabetical order from

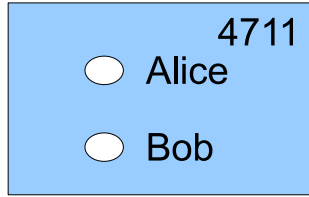


Fig. 1. Normal Ballot. A typical optical scan ballot configuration. Candidates are listed in a fixed order across all ballots and there is a designated location next to each candidate that a voter marks.

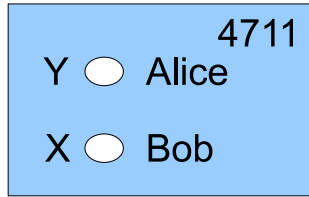


Fig. 2. New Ballot Configuration. Next to each oval there is a different symbol.

here onward. For each ballot, the symbols are shuffled relative to the *canonical order* and printed next to the ovals associated with the candidates as illustrated in Figure 2). Figure 3 illustrates the set of possible symbol orders for a two-candidate contest. Note that on any two different ballots, the symbols associated with the same candidate may be different. Therefore if the distribution of possible symbol orders is uniformly random across the ballots, the knowledge that a vote was cast for any particular symbol provides one with no statistical advantage in determining which candidate was voted for.

The serial number of each ballot is printed in such a way as to allow easy scanning. Assuming that scanner uses the mark sense technology, the serial number is represented using a matrix of digits, where the digits of the serial number blackened out. This is illustrated in Figure 4. Using this representation, any existing mark sense scanner can be configured to recognize a serial number even if its ignorant of the fact its scanning a serial number as opposed to votes. Mark sense attempts to determine if the geometric shape is filled in or not. Sets of shapes can be defined such that only one shape is allowed to be filled in, as would be done in for each contest on a first-past-the-post ballot, and if it more than one appears to be filled, the darkest is typically selected. The output of the scanner is an *electronic ballot images (EBIs)* which is a list of each shape and its state—filled or unfilled. For our method, the information stored by the precinct scanners will be the positions that were filled in, where some of these positions represent the serial number.

The last aspect of the new ballot is the reserved a portion that is to be used as a stub—detachable along a perforation in the paper. The stub also bares the serial number of the ballot printed on it but this serial number is for the voter to

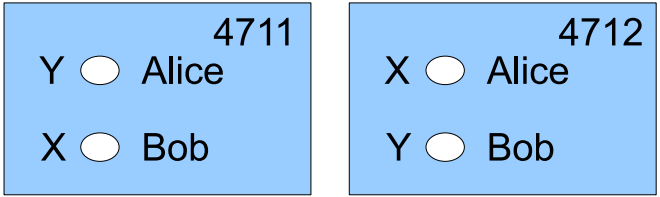


Fig. 3. Different Ballots. On two different ballots, the order of the symbols associated with the candidates may be different.

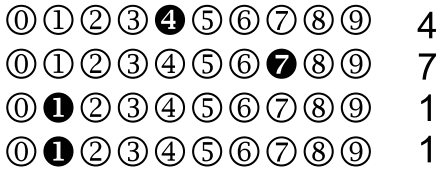


Fig. 4. Representing the serial number. This format allows a mark sense scanner to read it by checking which digits are black.

read, not the scanner, and can be represented in numeric form. A ballot adapted from a real election is shown in Figure 5. We now address the purpose of this stub and how it is used by the voters.

5 The Voting Ceremony

On voting day, after proper identification and verification, the voter is issued our modified ballot by the poll workers. She fills out this ballot as she would the ballot of any optical-scan system, filling in the ovals beside the candidates she wishes to vote for. Before she scans the ballot, she detaches the ballot stub and may write on it or any piece of paper the symbols that corresponded to the candidates she chose. Afterward, the ballot is scanned and placed into a ballot box as under the normal procedures of optical-scan voting.

While in the booth, if the voter makes a mistake while filling in her ballot, the ballot is officially marked as spoiled and given back to the voter. Also, the voter may choose to deliberately spoil a ballot in addition to the ballot she votes on. If she wishes to do so, she must tell the election official before getting her ballot. The election official will then present the voter with two ballots, both faced down, such that the voter cannot see the order of the symbols. The voter may choose one of the ballots to spoil and the other one to vote on. In section 6, we describe how these spoiled ballots can be used to audit the integrity of the election. The voter may keep the spoiled ballot for herself or she may choose to give them to an organization that she trusts (e.g. The League of Women Voters) to use in checking the integrity of the election on her behalf.

**SAMPLE OFFICIAL BALLOT
GENERAL ELECTION
POLK COUNTY, FLORIDA
NOVEMBER 7, 2000**

PRESIDENTIAL	CONGRESSIONAL	COUNTY																																																		
<p>ELECTORS FOR PRESIDENT AND VICE PRESIDENT (A vote for the candidates will actually be a vote for their electors.)</p> <p style="text-align: center;">(Vote for One Group)</p> <p>REPUBLICAN</p> <p><input type="radio"/> C GEORGE W. BUSH DICK CHENEY</p> <p>DEMOCRATIC</p> <p><input type="radio"/> G AL GORE JOE LIEBERMAN</p> <p>LIBERTARIAN</p> <p><input type="radio"/> B HARRY BROWNE ART OLIVER</p> <p>GREEN</p> <p><input type="radio"/> H RALPH NADER WINONA LaDUKE</p> <p>SOCIALIST WORKERS</p> <p><input type="radio"/> A JAMES HARRIS MARGARET TROWE</p> <p>NATURAL LAW</p> <p><input type="radio"/> E JOHN HAGELIN NAT GOLDBABER</p> <p>REFORM</p> <p><input type="radio"/> I PAT BUCHANAN EZOLA FOSTER</p> <p>SOCIALIST</p> <p><input type="radio"/> J DAVID McREYNOLDS MARY CAL HOLLIS</p> <p>CONSTITUTION</p> <p><input type="radio"/> F HOWARD PHILLIPS J. CURTIS FRAZIER</p> <p>WORKERS WORLD</p> <p><input type="radio"/> K MONICA MOOREHEAD GLORIA LA RIVA</p> <p><input type="radio"/> D _____ Write-in For President/Vice President</p>	<p>UNITED STATES SENATOR (Vote For One)</p> <p><input type="radio"/> L BILL McCOLLUM REP <input type="radio"/> R BILL NELSON DEM <input type="radio"/> P JOE SIMONETTA LAW <input type="radio"/> O JOEL DECKARD REF <input type="radio"/> S WILLIE LOGAN NPA <input type="radio"/> T ANDY MARTIN NPA <input type="radio"/> M DARRELL L. McCORMICK NPA <input type="radio"/> N _____ Write-in</p> <p>REPRESENTATIVE IN CONGRESS 15TH CONGRESSIONAL DIST. (Vote For One)</p> <p><input type="radio"/> W DAVE WELDON REP <input type="radio"/> Y PATSY ANN KURTH DEM <input type="radio"/> X GERRY L. NEWBY NPA <input type="radio"/> U _____ Write-in</p> <p style="text-align: center;">STATE</p> <p>TREASURER (Vote For One)</p> <p><input type="radio"/> B TOM GALLAGHER REP <input type="radio"/> A JOHN COSGROVE DEM</p> <p>COMMISSIONER OF EDUCATION (Vote For One)</p> <p><input type="radio"/> E CHARLIE CRIST REP <input type="radio"/> C GEORGE H. SHELDON DEM <input type="radio"/> D VASSILIA GAZETAS NPA</p> <p style="text-align: center;">LEGISLATIVE</p> <p>STATE REPRESENTATIVE 44TH HOUSE DISTRICT (Vote For One)</p> <p><input type="radio"/> F DAVE RUSSELL REP <input type="radio"/> G GREGORY L. WILLIAMS DEM</p>	<p>SHERIFF (Vote For One)</p> <p><input type="radio"/> I LAWRENCE W. CROW, JR. REP <input type="radio"/> H KIRK WARREN DEM</p> <p>SUPERINTENDENT OF SCHOOLS (Vote For One)</p> <p><input type="radio"/> K JIM THORNHILL REP <input type="radio"/> J DENNY DUNN DEM</p> <p>COUNTY COMMISSIONER DISTRICT 1 (Vote For One)</p> <p><input type="radio"/> M DON GIFFORD REP <input type="radio"/> L JANET SHEARER DEM</p> <p style="text-align: center;">COUNTY - NONPARTISAN</p> <p>SUPERVISOR OF ELECTIONS (Vote For One)</p> <p><input type="radio"/> O LORI EDWARDS <input type="radio"/> N BARBARA OSTHOFF</p> <div style="text-align: center; margin-top: 10px;"> <table style="border: none;"> <tr><td>■</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>■</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>■</td><td>8</td><td>9</td></tr> <tr><td>0</td><td>■</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr> <tr><td>0</td><td>■</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr> </table> </div>	■	1	2	3	4	5	6	7	8	9	0	1	2	3	■	5	6	7	8	9	0	1	2	3	4	5	6	■	8	9	0	■	2	3	4	5	6	7	8	9	0	■	2	3	4	5	6	7	8	9
■	1	2	3	4	5	6	7	8	9																																											
0	1	2	3	■	5	6	7	8	9																																											
0	1	2	3	4	5	6	■	8	9																																											
0	■	2	3	4	5	6	7	8	9																																											
0	■	2	3	4	5	6	7	8	9																																											



Fig. 5. Ballot adapted from Florida’s 2000 Polk County Election [pol07]. Attached to a ballot there is a stub with the serial number of the ballot printed on it.

After the polls close, anyone can go to the *bulletin board*—an official election web site—and enter the serial number of the ballot from a ballot stub. The *bulletin board* responds with a list of symbols corresponding to the symbols that were paired with the chosen candidates on the ballot.

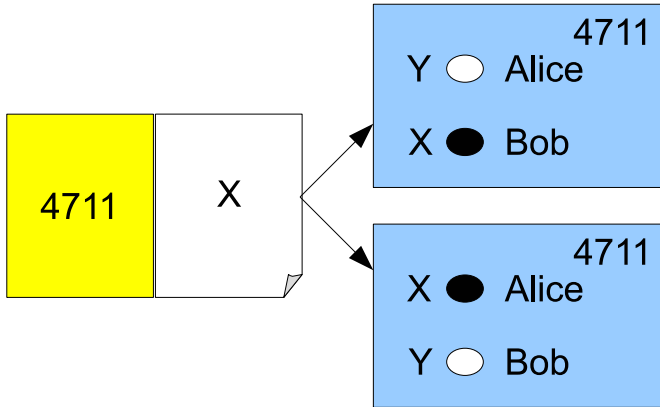


Fig. 6. Plausible Deniability. Knowing which symbol was marked does not reveal which candidate was chosen, since on different ballots the same symbol can correspond to different candidates.

Since the same symbol corresponds to different candidates on different ballots, the *bulletin board* preserves ballot secrecy while providing enough information to verifiably calculate election results. For example, if a coercer attempts to influence a voter to cast a vote for “Alice” and sees on the bulletin board that the ballot with her serial number had the oval corresponding to symbol “X” marked, symbol “X” may correspond to “Alice” or “Bob” as shown in Figure 6. Thus, the coercer has no assurance that the voter followed his directions. Even if she wanted to, the voter cannot provide any evidence that any symbol corresponds to a certain candidate for a ballot, since the paper she used to write down the symbols is merely a personal record and does not bear any value for anyone else but the voter.

If, after checking the *bulletin board*, the voter sees the same symbols she recorded on her piece of paper, she is guaranteed the following three things:

1. The scanner properly read her ballot.
2. The scanner correctly posted the symbol.
3. The central election authority received and counted a ballot with the correct symbol.

More generally, the voter is given assurance that her vote was properly interpreted and received by the election authority. However she still does not know if her ballot was counted correctly for candidate she chose. The next section presents the mechanism that ensures everyone, not only voters, that all the votes were counted as recorded—*i.e.*, that for each given serial number, the symbols that are posted on the bulletin board are counted as votes for the same candidate that the symbols appeared beside on that ballot.

6 Generation of the Election Results

In the previous sections, we have described a new front-end for the Punchscan voting system. We give a brief, conceptual overview of Punchscan's back-end, and then address how it is used to produce a universally verifiable set of election results.

6.1 Punchscan's Back-End: The Punchboard

The back-end of the Punchscan system, which translates voter marks into voter choices, is called the Punchboard, and it is a specially constructed anonymity network. A variety of anonymity networks have been proposed in literature, most of which are based on Chaum's mix networks [Cha81]. In a mix network, anonymous messages are encrypted multiple times (forming an 'onion') and then sent through a series of nodes, each of which remove the order-based correspondence between their input and output message set by using a secret permutation, and remove content-based correspondence by decrypting each message once. A second type of anonymity network, also due to Chaum, is the DC-net [Cha88] which hides an anonymous message amidst random numbers that cancel out when summed together, revealing the message. The Punchboard is both distinct from these two anonymity networks and similar in certain regards.

Similar to a mix network, the Punchboard operates on a batch of input messages (voter marks) and produces an output set (voter choices) in a permuted order. However unlike a mix network, the Punchboard does not distribute trust among multiple nodes. Instead the permutations are produced from a secret election-wide key that is shared among multiple trustees in a threshold scheme. This election key is never stored in memory but is regenerated by a cryptographic combination of trustee-supplied pass phrases on each occasion that the Punchboard is needed. The election trustees can be comprised of adversarial opponents, such as representatives of each political party, along with government officials to ensure the group has no incentive to collude.

Before the election commences, the election trustees use this election-wide key to generate the secret permutations in the Punchboard. A cryptographic commitment for each is computed and published. These commitments will later form the basis of a proof that the Punchboard has not been altered through the course of the election. This proof can be verified by any independent, interested party. There is one path per ballot for the election. An advantage of the Punchboard over the use of a mix network, which are used in some other voting systems [CRS04], is that the Punchboard does not use encryption and decryption functions. Rather it uses a much faster method based on modular addition. The voter marks can be thought of as the sum of some random numbers (represented on a Punchscan ballot by the random order of the symbols) and the position marked by the voter. The first node of the Punchboard adds an additional random number to this sum so it is untraceable if one were to examine the input and output set of the first node. The second node subtracts off the sum of all the random numbers leaving the position of the candidate to

receive a particular ballot's vote³. As with the secret shuffles, all the random numbers to be used are generated from the election-wide key and fixed prior to the election through a cryptographic commitment.

Assuming that randomized partial checking [JJR02] is used for auditing the Punchboard, a minimum of two nodes are sufficient. The commitments to the random numbers can be independent, or can be blended into the commitments to the paths, meaning that a path and a set of numbers are committed to at the same time, using a single commitment. The Punchboard is very fast because it can be implemented using an efficient hash-based commitment scheme and without the use of public key or symmetric key cryptography. Full details of the Punchboard have been omitted from this document but are available in previous publications [PH06, ECCP07].

6.2 Using the Punchboard with the New Ballots

To ensure that the ballots are printed properly, some printed ballots are checked in an audit. This audit, which is similar to the audit in [AN06], is performed by a *designated challenger*, who is a voter or representative of an independent organization at the poll site. The designated challenger chooses a ballot from the set of ballots. A poll worker then records the ballot as spoiled, marking all the contests on it so that it cannot be used in the election, and gives it to the designated challenger. For each spoiled ballot, the election authority reveals the entire path through the Punchboard. The designated challenger is then able to check the revealed path against what was committed to for that ballot, and be assured that each symbol paired with the proper candidate and therefore would have been properly counted. This process can be repeated until a predefined statistical certainty is reached that all the ballots are printed exactly as they were committed to.

In Punchscan, top layer symbols and bottom layer symbols are permuted from the canonical order according to two statistically independent permutations. In the new ballot style, the two permutations are composed such that marking a certain symbol on the new ballot is equivalent to marking a certain hole in Punchscan—*e.g.* marking the second symbol on the new ballot is equivalent to marking the second hole on a Punchscan ballot in terms of which candidate receives the vote. Using this method, the voter marks on the new ballot can be transformed into voter choices using a Punchboard identical to the ones used for Punchscan ballots. Indeed, this actually permits the new ballots and Punchscan ballots to be mixed within the same election, and the ballot styles can be made indistinguishable in their presentation on the bulletin board.

Since the optical scanner cannot read the symbols that were marked, it reads the candidates that were chosen on each ballot. After receiving the electronic ballot images from the polling place scanners, the election authority transforms these clear votes into marked symbols. The information required to perform this transformation is stored on the Punchboard. After this step, the symbols and the results are posted on the official web site.

³ Instead of numbers in Z_n and modular addition, permutations and permutation composition can be used, or more general any elements in a group and the group operation.

7 Dispute Resolution

Any *challenger* in possession of a ballot stub, usually a voter, may initiate the dispute resolution process if she believes the record on the *bulletin board* is incorrect. This process provides a privacy preserving method to resolve discrepancies between voter records and the *bulletin board*. The reason records may not be the same are usually caused by two situations:

1. **The Voter is Wrong.** Because there is no control over what symbol the voter records, she could have recorded a symbol other than the symbol paired with her chosen candidate.
2. **The Record is Wrong.** A scanner or software error may cause the recorded vote to be incorrect. It is also possible that an attacker may have changed or altered ballots, or that some other malfeasance occurred.

In 1, dispute resolution convinces all *challengers* that each ballot was received, unaltered, and counted correctly. In 2, dispute resolution provides proof to any observer that the record on the *bulletin board* is correct. In both situations, dispute resolutions preserves ballot privacy.

The dispute resolution protocol is carried out between the election *official* and a set of *challengers* as follows:

1. **Proving the Ballot is Present and Unaltered**
 - (a) *Challengers* present ballot stubs.
 - (b) *Official* retrieves ballots with serial number from each stub and places them into a privacy sleeve that does not reveal candidate choices, but does show the rest of the ballot including the part of the ballot where the stub was taken and the back of the ballot. It should also show candidate lists, but should hide the choices and symbols.
 - (c) *Challengers* verify ballot sheet has not been modified and may conduct forensic analysis to verify the ballot stub was at one point attached to the ballot presented by the *official*.
2. **Showing the Selected Ballot Letters**
 - (a) For each ballot with the same letter marked, the *Official* moves the ballot to a separate privacy sleeve being careful to hide the choices made on the ballot (e.g. with the back of the ballot facing the *challengers*). This sleeve does not show the serial number but does show the choices and symbols of the race in dispute. *Official* then drops each ballot into an empty lottery-style hopper⁴.
 - (b) All ballots with the same symbol marked are mixed, and removed from the hopper.
 - (c) Each *challenger* verifies that all ballots have the same symbol next to the chosen candidate.
 - (d) This process repeats for all sets of ballots with different symbols marked.

⁴ If not enough ballots are available, the *official* can add fake ballots to the hopper.

After all disputes are settled, everyone can assume that the public record of chosen symbols is correct, and that no ballots were lost. If necessary, officials re-compute the results from the corrected public record.

The current method may be unable to ballot secrecy if the race in question is too long or not enough ballots with the same symbol marked are challenged. Finding a more efficient dispute resolution procedure that does not require physical interaction or forensic analysis is an open problem to be addressed in future work, as are the best methods for dealing with ballots with write-ins.

8 Implementation

Given that the speed of the system is largely dependent on the speed of the Punchboard, the performance of our method is nearly equivalent to that of Punchscan. However, we have produced an implementation and tested with both moderate and large-sized elections. On a 1.73 GHz laptop, we were able to tabulate 1 million ballots in under 10 minutes. Using actual statistics from Florida's 2000 Polk County election as a benchmark [pol07], where there were 32 contests with an average of 3.2 candidates per contest, we tabulated 200,000 ballots in under 4 minutes and audited the Punchboard in less than 2 minutes.

The complete source code written in Java, as well as object code, for our implementation is available [web07], along with instructions on how to build it and use it in a mock election. The object code is directly accessible from any browser, via Java Network Launching Protocol (JNLP), without installing any of the cryptographic libraries our implementation depends on or performing any other specific configuration.

Future work in this area includes developing an easier and more efficient dispute resolution process or minimizing its need by using other techniques, procedures for limiting access to the ballots before and after they are handed out to the voters, addressing forced randomization attacks and simplifying the Punchboard for the sake of efficiency and explanatory ease, especially with individuals unfamiliar with verifiable mix networks or other anonymity networks.

9 Concluding Remarks

In this paper, we have created a new front-end for Punchscan with numerous improvements. The new ballots eliminate the indirection in order to increase usability. They are printed on a single sheet, which allows more efficient use of resources, and the sheets do not contain holes which allows for on-demand printing. The ballots are filled out exactly as in optical-scan voting which should be a method familiar to many voters in the US. However, the greatest advantage of our method is that it can be used as an add-on to the optical-voting systems already owned by many election districts without software upgrades. Our method does not interfere with the tally produced by optical-scan equipment, nor does it preclude the option of a manual recount. Without detracting in any way, we add end-to-end verifiability to a popular voting system and provide unconditional assurance that every vote was counted accurately.

References

- [AN06] Adida, B., Andrew Neff, C.: Ballot casting assurance. In: EVT 2006: Proceedings of the USENIX/Accurate Electronic Voting Technology Workshop 2006 on Electronic Voting Technology Workshop, Berkeley, CA, USA, p. 7. USENIX Association (2006)
- [AR06] Adida, B., Rivest, R.L.: Scratch & vote: self-contained paper-based cryptographic voting. In: WPES 2006: Proceedings of the 5th ACM workshop on Privacy in electronic society, pp. 29–40. ACM Press, New York (2006)
- [Ben07] Benaloh, J.: Ballot casting assurance via voter-initiated poll station auditing. In: Preproceedings of the 2007 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT 2007) (August 2007)
- [CEC+08] Chaum, D., Essex, A., Carback, R., Clark, J., Popoveniuc, S., Sherman, A., Vora, P.: Scantegrity: End-to-end voter-verifiable optical-scan voting. *IEEE Security & Privacy* 6(3), 40–46 (2008)
- [Cha81] Chaum, D.L.: Untraceable electronic mail, return address, and digital pseudonym. *Communication of ACM* (February 1981)
- [Cha88] Chaum, D.: The dining cryptographers problem: unconditional sender and recipient untraceability. *J. Cryptol.* 1(1), 65–75 (1988)
- [CRS04] Chaum, D., Ryan, P.Y.A., Schneider, S.A.: A Practical, Voter-verifiable, Election Scheme. Technical Report Series CS-TR-880, University of Newcastle Upon Tyne, School of Computer Science (December 2004)
- [ea07] Bowen, D., et al.: California secretary of state, voting systems review: Top-to-bottom review (2007), http://www.sos.ca.gov/elections/elections_vsr.htm
- [ECCP07] Essex, A., Clark, J., Carback, R.T., Popoveniuc, S.: The Punchscan voting system: VoComp competition submission. In: Proceedings of the First University Voting Systems Competition (VoComp) (2007)
- [EDS06] Election Data Services. 2006 voting equipment study (October 2006)
- [JJR02] Jakobsson, M., Juels, A., Rivest, R.L.: Making mix nets robust for electronic voting by randomized partial checking. In: Proceedings of the 11th USENIX Security Symposium, Berkeley, CA, USA, pp. 339–353. USENIX Association (2002)
- [KMRS06] Kiayias, A., Michel, L., Russell, A., Shvartsman, A.A.: Security assessment of the diebold optical scan voting terminal (October 2006)
- [Nef04a] Andrew Neff, C.: Practical high certainty intent verification for encrypted votes (October 2004)
- [Nef04b] Andrew Neff, C.: Verifiable mixing (shuffling) of el-gamal pairs (October 2004)
- [PH06] Popoveniuc, S., Hosp, B.: An introduction to PunchScan. In: IAVoSS Workshop On Trustworthy Elections (WOTE 2006), Robinson College, Cambridge UK (June 2006); Also GWU-CS Technical Report
- [pol07] Election Website for Polk County, Florida (October 2007), <http://www.polkelections.com/>
- [web07] Punchscan voting system website (October 2007), <http://www.punchscan.org/>
- [WJB06] Wagner, D., Jefferson, D., Bishop, M.: Security analysis of the diebold accubasic interpreter (February 2006)

Attacking Paper-Based E2E Voting Systems

John Kelsey¹, Andrew Regenscheid¹, Tal Moran², and David Chaum³

¹ National Institute of Standards and Technology
{john.kelsey, andrew.regenscheid}@nist.gov

² Harvard SEAS Institute of Science

talm@seas.harvard.edu

³ info@chaum.com

Abstract. In this paper, we develop methods for constructing vote-buying/coercion attacks on end-to-end voting systems, and describe vote-buying/coercion attacks on three proposed end-to-end voting systems: Punchscan, Prêt-à-voter, and ThreeBallot. We also demonstrate a different attack on Punchscan, which could permit corrupt election officials to change votes without detection in some cases. Additionally, we consider some generic attacks on end-to-end voting systems.

1 Introduction

Voting systems in widespread use today have a number of known vulnerabilities [1,2,3]. Many of these vulnerabilities can be mitigated by following certain procedures; the integrity of the election is then dependent on a combination of correct behavior by software, hardware, and election officials.

The best of these systems provide security assurance based on the honesty and correct behavior of a small set of election officials and other observers. Commonly, each political party or candidate provides a certain number of observers. These individuals are expected to notice and report fraud that would deprive their party or candidate of votes. Election officials are also expected to notice and report fraud. In general, an outsider attempting to decide whether to trust a reported election outcome must rely on the premise that correct procedures were followed by observers and election officials.

A new kind of voting system has been proposed in recent years [4,5,6,7,8,9], in which the voter interacts with the voting system to get a *receipt*. This receipt can then be checked against a set of receipts published by the voting system. These published receipts can be used to produce or verify the reported count for the voting system, but must not be useful for selling votes (for example, by proving how each voter voted). This class of voting system has been called *end to end*, or *E2E*, to reflect the idea that each voter can check, to some high degree of confidence, that his vote was correctly cast, and also that his vote was correctly included in the final count. Other means must be used to ensure that the whole election result is correct- for example, by ensuring that only authorized people were permitted to vote, and that no additional votes were inserted into the count. These systems build on older work on cryptographic voting systems [10,11,12],

which typically relied on the assumption that a voter would have access to some trusted computing devices.

In this paper, we consider the security of a number of proposed E2E voting systems against attacks to tamper with election results and to permit the buying or coercion of votes.

An important idea in this paper is that most E2E systems can be meaningfully divided into:

1. A *front-end*, which describes the voter's direct interaction with the voting system to cast a vote and receive a receipt, and
2. A *back-end*, which describes the voting system's public statements (such as receipts posted to a bulletin board and the claimed vote totals), and the mechanisms used to prove to voters and other observers that the reported election results are consistent with the public statements.

Our attacks focus exclusively on the front-end, the voting systems' interactions with the voters. In general, our attacks provide ways in which corrupt election officials or voters can subvert the real-world implementation of the front-ends of these voting systems, to get very different properties from the voting systems than were expected.

1.1 Attacking Voting Systems

In general, someone attacking a voting system wants to affect the outcome of the election. The stakes in such an attack can be quite high, involving control of enormous government resources. These stakes can be inferred by the amount of money spent on lobbying and campaigning, both reported publicly in the United States [13,14].

Election results can be changed by altering recorded votes or reported totals, and also by finding a way to learn what each voter chose, so that voters can be bribed or coerced into voting in some desired way. It can also be done by disrupting the orderly operation of an election, which may simply delay an undesirable (to the attacker) result, or may force an election to be rerun, possibly changing its result. Violations of voter privacy and disruption of elections may be of some interest to attackers even when the election result cannot be altered, but the impact of these attacks is much smaller.

1.2 Previous Work

Several end-to-end cryptographic voting protocols have been developed. This paper analyzes Punchscan [6,7], Prêt-à-voter [15] and ThreeBallot [8,16], which are described in Section 2.

Researchers have begun to perform security analyses of these schemes, and some weaknesses have been discovered. A coercion attack against ThreeBallot, dubbed the ThreePattern attack [8], involves a coercer telling voters to mark their three ballots according to a particular pattern, then checking that that

those patterns appear on the bulletin board. Strauss [17,18] notes several vulnerabilities in ThreeBallot, including the Reconstruction attack. Here, for sufficiently long ballots, an attacker is able to look at one receipt and determine which other two ballots on the bulletin board belong to the same multiballot.

A well known issue with these systems is that it is easy to force a voter to vote for a random candidate by instructing them to return with a receipt showing a vote for the first ballot choice. A formal study of the three voting schemes by Clark, Essex and Adams [19] concludes that while ThreeBallot receipts provide some clues for how voters voted, Prêt-à-voter and Punchscan receipts do not contain any information that would help an attacker. Nonetheless, Moran and Naor [20] developed a coercion attack against Punchscan that relied on the voter's choice of receipt. The Punchscan voting procedure was modified for the VoComp competition [21] to prevent this and related attacks by requiring voters to choose the receipt sheet prior to viewing the ballot.

1.3 Our Results

Briefly, our results can be summarized as follows:

Punchscan and Prêt-à-voter

- We provide a misprinting attack which can alter election results by misleading many voters into believing they have a receipt committing to a different vote than is actually cast. This can be mitigated with a special audit of the printed ballots, but that auditing requires trusting small numbers of election observers, rather than all voters, with the integrity of the election.
- We provide a mechanism for using scratch-off cards, cellphones, or other techniques to reliably buy or coerce votes.

Prêt-à-voter

- We report a previously-known but unpublished sleight-of-hand attack to allow vote buying.

Threeballot

- We provide a mechanism to provide voters a financial incentive to vote in some desired way, when the three ballots are filled-in in a random way by some voting machine.
- We provide a technique for buying votes when the ballots are filled out manually, using a variant of chain voting.

All End to End Systems

- We report a couple of known broad categories of attack on E2E systems we did not find referenced in the literature.
- We provide a framework for vote-buying and coercion attacks.

While our attacks are specific to particular E2E systems, the general ideas behind them can be broadly applied. One goal of this paper is to get these ideas into widespread circulation, so that systems we have not considered here may also be subjected to the same analysis.

2 Background

2.1 E2E Voting Systems

Election systems in current use rely on procedures to provide integrity and ballot secrecy. Typically, voters must trust election administrators to follow these procedures. Secure elections using traditional voting systems are possible when tight controls are in place, such as maintaining the chain of custody of ballots, but it is nearly impossible for voters to gain assurance that such controls are followed. End-to-end (E2E) cryptographic voting schemes aim to provide voters with a means of verifying that elections are honest, without needing to trust election officials or that the chain of custody of ballots is maintained.

End-to-end refers to the voter's ability to verify the election from vote casting to vote counting. Most schemes operate by encoding voters' choices a special way which can be read by election officials. The encoded ballot is posted on a public bulletin board and each voter is given a voting receipt. The unique feature of E2E voting schemes is that this receipt can be used to verify the encoded ballot on the bulletin board but does not show how the voter voted.

Additionally, E2E voting schemes provide voters with a means to ensure cast ballots are counted correctly. In nearly all cases this done by having the voting scheme prove that each encrypted vote on the bulletin board is correctly decrypted, allowing anyone to verify the final vote tallies by recounting the decrypted ballots.

Front-End vs. Back-End. E2E voting schemes involve a combination of activities performed by voters, election administrators and auditors. We refer to the part of the voting system that the voter interacts with the system as the *front-end*. This typically includes the ballot, receipt and bulletin board. Voters interact with the front-end to gain assurance the voting system is functioning honestly, often relying on auditors or tools to verify some parts of the voting protocol for them. The voting scheme back-end is everything that occurs partially hidden from the voter. This can include the cryptographic encoding and decodings of ballots, ballot shuffling and various third-party auditing techniques. The attacks discussed in this paper take place within the front-end of voting schemes. They involve presenting misleading information to voters that cause their votes to be miscounted, or providing voters with specific ways to interact with the front-end of the voting scheme which can be used to encourage or coerce particular votes.

2.2 Punchscan

Punchscan [6,7] is a paper/electronic hybrid cryptographic voting scheme that uses paper ballots.¹ Each Punchscan ballot consists of two separate sheets. Voters must interact with both of these sheets to cast a vote. Viewed separately, neither sheet directly contains sufficient information to determine the selections of the voter.

The top sheet of a Punchscan ballot contains the set of ballot questions. For each question, the ballot maps each choice to a particular letter (or other symbol), along with a set of holes. Those holes line up with a permutation of the set letters which is printed on the bottom sheet. When the top sheet is stacked directly on top of the bottom sheet the voter sees each question, mappings from choices to letters, and the letter options through the holes in the top sheet.

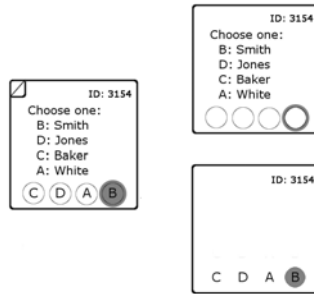


Fig. 1. A Punchscan Ballot with vote for Smith

To cast a vote, the voter looks up the letter corresponding to his or her desired response, and finds that letter by looking through the holes in the top sheet. The voter then applies a bingo dauber to that letter, thereby marking the letter on the bottom sheet and the area around the hole on the top sheet. The two sheets of the ballot are separated and the voter selects one to destroy. The remaining sheet is scanned, providing election administrators with a digital record of the vote, while physical sheet becomes the voter's receipt.² A representation of the digital ballot is posted on a public bulletin board, allowing the voter to compare the marks on the receipt to those appearing on the bulletin board.

Election administrators use a table containing directions for decrypting each ballot sheet to tally votes, called the Punchboard. By conducting audits before and after the election, voters can be assured that their ballots halves are correctly translated into their desired votes. The details of these audit procedures contain

¹ Recently, a related set of end-to-end voting systems have been developed under the name Scantegrity. We do not consider Scantegrity in this paper, but the mechanisms are different enough that our current attacks do not appear to apply.

² The current Punchscan voting procedure requires that voters select the top or bottom sheet as the receipt prior to viewing the ballot. In this paper we will propose attacks against both sets of Punchscan election procedures.

the bulk of the cryptographic techniques. However, rather than attacking the underlying cryptographic primitives of the election system, we will attack the voting procedure.

2.3 Prêt-à-voter

A detailed description of the Prêt-à-voter scheme can be found in [15]. Like Punchscan, Prêt-à-voter encodes votes based on a random permutation, in this case of the candidates. A Prêt-à-voter ballot is split between two halves, separated by a perforated edge. The left half of the ballot displays the candidates in a permuted order, while the right half has boxes that are marked to indicate a vote. Also, the right half contains a cryptographically-protected copy of the permutation of the candidates on the left side. This permutation could be encrypted using threshold cryptography or onion encryption, so only a group of election administrators would be able to decrypt that permutation. Typically, the permutation of the candidates is a cyclic shift, represented as an offset from a standard candidate ordering.

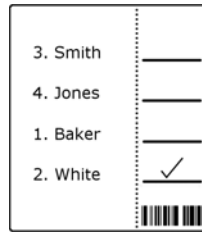


Fig. 2. A Prêt-à-voter Ballot with vote for White

To cast a ballot, voters mark the box next to the candidate of their choosing, and separate the two halves along a perforated edge. The halves containing the candidates names are destroyed, while the halves with ballot marks are scanned in and recorded as votes. Voters are allowed to bring home the right halves of ballots and compare them to the scanned copies, which are posted to a bulletin board. The cryptographic algorithms in the scheme allows voters to verify that the collection of posted ballots is properly decrypted.

The Prêt-à-voter system relies on the proper construction of ballots. That is, the right half of the ballot must contain an encrypted representation of the permutation of candidates displayed on the left ballot half. Each voter may choose to audit a ballot by providing a special auditing device with the just the right ballot half. The device would decrypt that half and respond with the permutation of candidates. The voter could verify that this permutation matches the candidate list on the left ballot half. Auditing a ballot also invalidates that ballot, forcing a voter to obtain another ballot to either cast or audit. The attacks outlined in this paper rely on the voter being allowed to choose between casting or auditing a ballot after viewing it.

2.4 ThreeBallot

The ThreeBallot voting system [8], like Punchscan and Prêt-à-voter, uses paper-ballots. Unlike those systems, however, ThreeBallot is entirely paper based without requiring advanced cryptographic techniques to perform auditing or maintain voter privacy.

In ThreeBallot, each voter receives three identical ballots, each with a unique serial number. To vote for a particular candidate, a voter marks the candidate's name on exactly two of the three ballots. For each of the remaining candidates, the voter marks the candidate's name on exactly one of the ballots. The voter then feeds the multi-ballot into a checker which verifies that the ballot has been properly filled out. If so, the checker places a red strip across the multi-ballot and asks the voter to choose one of the three ballots to copy. The checker separates the ballots and returns them to the voter, along with a copy of the chosen ballot. The voter casts the original three ballots in a ballot box and takes the copied ballot home as a receipt.

After the election digital representations of the cast ballots are posted on a bulletin board. The voter can verify that there is a ballot on the bulletin which matches the receipt taken home. Anyone can also tally the results of the election by merely counting the votes on the ballots. The resulting tallies will be inflated by the number of voters in the election.

150423	892314	239782
Smith ○	Smith ●	Smith ○
Jones ○	Jones ●	Jones ●
Baker ●	Baker ○	Baker ○
White ○	White ●	White ○

Fig. 3. A ThreeBallot Ballot with vote for Jones

To improve the usability of ThreeBallot, it has been suggested [8] that voters could interact with an electronic ballot marker (EBM) that would provide an interface similar to that of a DRE. After the voter selected her choices, the EBM would print out a randomly filled-in multiballot that would correspond to those choices. The voter could verify that the multiballot properly reflected her intended vote, and obtain a receipt for any one of the three ballots.

3 Election Fraud with Misprinted Ballots

The most serious threat to an election is an attack capable of changing the outcome of the election. The goal of E2E schemes is to make any changes detectable. Most E2E schemes claim to provide this, but such claims are only

supported when election officials and auditors can be trusted to honestly follow proper election procedures.

The accuracy of vote counts in Punchscan and Prêt-à-voter is dependent on the proper construction of ballots. To deal with this both systems rely on pre-election audits of the ballots to ensure the ballots were created correctly. In the Punchscan scheme, election officials commit to the set of ballot forms and audit some percentage of the ballots, looking for irregularities between the actual ballot forms and the commitment on the Punchboard. Tampering with the set of ballots between the audit and the election has a good chance of being caught during the post-election audit. Here we will describe a way to tamper with the ballots in a way that would not be detected with typical audit procedures.

In this attack, a small percentage of ballots are replaced with tampered ballots. The front sheet of each of these ballots remains the same as their untampered versions, while the back sheet is changed such that the placement of two letters are swapped. Figure 4 gives an example of a tampered ballot. In this configuration, votes for Smith and Jones will be swapped. That is, a voter attempting to vote for Smith would mark the third hole, but that vote would instead be decrypted by the Punchboard to be a vote for Jones. Note that an attacker could alternatively misprint the front sheets, keeping the back sheets untampered.

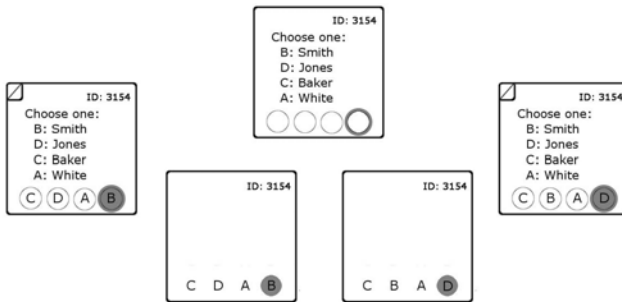


Fig. 4. Flipping Two Letters On Back Sheet

Misprinted ballot sheets are easily identifiable when compared to the Punchboard, as they will not match the committed ballots. Misprinted back sheets, when kept as a receipt, would provide evidence of election tampering. However, when the front sheet is kept as a receipt, the back sheet, the only evidence of tampering, is destroyed during the voting process. This leads to a very simple vote swapping attack when voters tell election officials what sheets they will take as receipts before the officials hand out ballots. In this case, election officials can provide misprinted ballots to only those voters who choose to keep the unmodified front sheets. This could greatly change election results if voters of a particular political party are targeted for attack.

This attack could be prevented by forcing election officials to commit to a particular ballot before asking the voter to choose a receipt sheet. However, similar attacks are still possible. In this case, attackers need to ensure that voters

cannot cast a ballot and retain the modified sheet. This could be accomplished by constructing the modified back sheet such that it looks normal to the human eye but is unreadable to the scanner. Alternatively, the scanner could be involved in the attack, and recognize the ballot ID as a tampered rear ballot that should be rejected. In both cases the voter would have to obtain a new ballot and revote. This attack can be prevented by including special procedures for the handling of unscannable ballots. Unscannable ballots should be treated differently than other invalid ballots. Voters must be allowed to keep an unmarked copy of the unscannable ballot as a receipt and then be given an opportunity to vote again. During the post-election audit, election officials should open the commitment to the same half of the ballot the voter was given. The voter could use the receipt to check whether this attack occurred by comparing the opened commitment to the ballot receipt. The commitment to the other half should remain unopened in order to protect against coercion attacks.

Variations of this attack can be applied to other E2E schemes. A similar attack can be applied to Prêt-à-voter with the cooperation of ballot scanners. However, this attack appears much less likely to go unnoticed in practice. In this case, the right-hand side of the ballot would be altered by swapping the names of two candidates. Attackers would also modify ballot scanners so that they would respond with an error when modified ballots are audited.

4 Incentives, Voter Coercion and Vote Selling

Voter coercion attacks aim to give one person undue influence over another person's vote. Often, the attacker influences the voter by rewarding the voter for voting for a particular candidate or punishing the voter for failing to do so. However, as attackers can also seek to influence votes rather than force them, it is acceptable for coercion attacks to provide an incentive toward voting a particular way. This observation opens the door for a wide variety of attacks that, while not perfect, can be effective at influencing voters.

Voter coercion is inherently a contract. Voters agree to vote a particular way in exchange for a reward or to avoid punishment. A successful attack requires a way to enforce the contract. The goal of a coercion attack is to create a protocol between a coercer and voter that tends to reward voters that vote correctly. A protocol need not be perfect. It may occasionally reward voters incorrectly that voted for the wrong candidate. As long as the probability of being rewarded is higher when a voter votes correctly, the protocol will provide an incentive for voters to vote correctly. Alternatively, a protocol may occasionally fail to reward voters that vote correctly. Protocols that never fail to reward honest voters can be considered contract enforcement protocols. In this case, all voters failing to be rewarded by the protocol could be punished severely³, knowing that only dishonest voters would be punished.

³ A real world example of such punishment comes from the days of machine politics in the United States, where city or state employees' jobs could depend on voting the right way.

The voting system and the rules for what voters are allowed to bring into the voting booth together determine what components an attacker can use to enforce the vote buying protocol. For example, if voters are allowed to bring in cameras almost any voting system will fall to a vote buying attack. Nonetheless, it is impractical to ban everything from voting booths. A piece of paper and a pen, a pre-marked "voters guide" or even a cell phone might be used to enforce a vote-buying contract.

4.1 Forged Ballots

A well-known attack in the end-to-end cryptographic voting community involves providing voters forged ballot halves to destroy in place of actual ballot halves. Punchscan and Prêt-à-voter ballots are split between two halves. Combined these halves display a human-readable vote, but each half on its own acts as an encrypted vote that can only be read by the election officials. The combined sheets can show anyone how the voter voted. For that reason, the election procedures of Punchscan and Prêt-à-voter require that each voter destroy one half of the ballot and retain the other half.

Voters able to leave a polling place with both halves of the ballot can use these halves to prove how they voted. A voter may be able to do this without raising suspicion from the election officials by secretly bringing a forged ballot sheet to the polls. The voter would destroy the forged ballot sheet rather than one of the original sheets. For instance, a voter may bring in a copy of the front sheet of a Punchscan ballot. After voting, the voter would slip the actual front sheet in his pocket, destroy the forged sheet in front of an election official and keep the back sheet as a receipt⁴.

4.2 Incentives

Typical vote buying attacks involve an attacker paying individuals who prove that they voted for a particular candidate. However, resourceful attackers can still influence election results without learning how individuals voted by providing them with an incentive to vote a particular way. The idea is that voters will maximize their expected return by following the coercer's instructions. Three-Ballot, when used with an electronic ballot marker, is particularly vulnerable to incentive attacks⁵. In that variant of ThreeBallot, ballots are automatically marked by a machine. Voters make their selections on a DRE-like machine which randomly constructs a valid multiballot with votes for those selections. The attacks work on the principle that although voters cannot control the specific marks on their multiballots, their ballot choices will influence the marks.

⁴ Current Punchscan procedures include a clipboard lock. Each ballot is locked to a clipboard before being handed to a voter. After marking a ballot the voter tears one sheet out of the lock and destroys it, then returns to the official with the remaining sheet still locked in place. As voters, rather than election officials, destroy the ballot sheets, the clipboard locks do not prevent this attack

⁵ There are other attacks on hand-marked ThreeBallot which are not discussed here, notably the Italian attack.

ThreeBallot Pay-Per-Mark. A simple example of an incentive attack with a machined-marked ThreeBallot device is to pay voters for each mark on a receipt that is acceptable to the vote-buyer. For example, the vote-buyer would offer to pay one dollar for every mark corresponding to a member of the Whig party. A vote-seller would cast a multiballot, choose a receipt based on the number of Whig votes contained on each ballot and present that receipt to the buyer in exchange for payment. If the seller does not vote for the Whigs on any of n total questions, each ballot would contain roughly $\frac{n}{3}$ Whig marks. However, voters who vote for Whigs on every ballot question would expect to find a ballot with $\frac{2n}{3}$ Whig marks. This is ineffective at influencing individual races and does not work when races are separated from one another.

ThreeBallot Pay-for-Receipt. In many cases a vote buyer may only be interested in coercing a voter on a single ballot question. We can create an incentive in machine-marked ThreeBallot to encourage voters to vote for a particular candidate over another. Consider a close election between Smith and Jones. A vote-buyer attempting to gain votes for Smith could force voters to return with a ballot that contains a vote for Smith but not a vote for Jones. If a voter votes as directed, the voter is guaranteed to obtain a ballot that contains a vote for Smith but not Jones. However, if the voter instead casts a vote for Jones, then the voter only has a $\frac{1}{3}$ chance of obtaining such a ballot. If a neither Smith nor Jones is chosen, then the voter has a $\frac{2}{3}$ chance of obtaining such a receipt.

Because honest voters⁶ will always be able to return with the correct receipt this attack can also serve as a voter coercion attack, demanding that a voter return with the correct receipt to avoid punishment. Similar attacks can be conducted against Punchscan by extending the ideas in [20]. An attacker could develop a set of marked receipts, one of which is always obtainable if a voter votes as directed, but may not be if the voter votes for a different candidate.

150423	892314	239782
Smith ●	Smith ○	Smith ●
Jones ○	Jones ○	Jones ●
Baker ○	Baker ●	Baker ○
White ○	White ○	White ●

Fig. 5. Vote for Smith and Not Jones

Levels of Payment. We can construct slightly more complicated attacks that are effective against Punchscan. Moran and Naor present a simple coercion attack against a 2-candidate race in Punchscan in [20] which allows roughly $\frac{3}{4}$ of voters to vote how they wish, but forces $\frac{1}{4}$ of voters to vote for a particular

⁶ In this context, an "honest" voter is one who votes as he's told.

candidate. The attack works by paying for receipts marked particular ways; the ballot layout determines what the voter can do to get paid. We extend that approach here to work with multiple candidates. The basic idea is that we will pay people to vote against a particular candidate by using different levels of payouts for different receipts.

Consider a vote buyer who wants to see Smith lose an election with n candidates. The buyer would offer \$10 for any front receipt showing Smith= a with the first hole marked, or \$5 for any back receipt marked for a . If we assume voters will always act to maximize their payout, any voters receiving a ballot where Smith= a will return with the front sheet marked to randomize their vote. Thus, we know any voter returning with a marked on the back sheet did not vote for Smith. Effectively we are randomizing votes away from Smith. About $\frac{1}{n^2}$ of voters will vote for Smith, while $\frac{n+1}{n^2}$ of voters will vote for each of the remaining $n - 1$ candidates, assuming voters always act to maximize their payoff.

4.3 Scratch-Off Card Attacks

Two-way communication between a voter in the voting booth and an attacker can be a very powerful tool for creating coercion attacks. In this section we will present several coercion attacks on E2E systems that work by simulating two-way communication entirely within the voting booth. This is based on similar work by Moran and Naor in [22] that used scratch-off cards to construct polling protocols. By scratching off a portion of this card based on a marked ballot, voters will permanently bind themselves to that ballot. The scratch-off card provides a challenge to the voter, which they cannot receive until after committing to a ballot form.

Basic Idea. Here we will present a simplified vote selling attack where the vote buyer is in contact with a voter inside the voting booth using a cell phone. The cell phone provides a means of communicating challenges and pledges between the buyer and voter. Using the two-way communication with the vote buyer, and the receipt provided by the voting system, the voter is able to convince the buyer that he voted for the correct candidate.

The important observation here is that letting a voter choose one of two sheets to retain reveals as much information as letting the voter retain both sheets. This is damaging to paper-based E2E systems where a single ballot is split across multiple sheets. In the case of Punchscan, possession of one receipt and knowledge of the other (destroyed) sheet is sufficient to determine the voter's selection. This leads to the following vote-buying protocol:

1. The voter obtains a Punchscan ballot and enters booth.
2. Using the cell phone, the voter issues pledges for the two ballot sheets by telling the buyer the letters associated with each candidate (the contents of the top sheet) and the orders of the letters (the contents of the bottom sheet).
3. The buyer randomly selects one of the pledges sheets and issues this selection to the voter as a challenge.

4. The voter keeps the challenged sheet as a receipt, casts the ballot, and returns to the buyer.
5. The buyer compares the receipt to the pledged sheets from Step 2.

This protocol acts as a cut-and-choose proof of the truthfulness of the voter's pledges. Now that the buyer has obtained one sheet, and is convinced of the contents of the other sheet, it is easy to determine the vote cast.

Scratch-Off Card. It might be difficult to have a cell phone conversation in the voting booth without being noticed by an election official. In this section we will discuss how to run the receipt-based vote buying protocol using a scratch-off card in place of cell phone communication. We can replace the cell communication with anything that lets a voter pledge commitments to two ballot sheets, and only then receive a challenge.

Here we will show how this can be accomplished using scratch-off cards. Suppose a group of voters agree to sell their votes to Smith. A voter can commit to the two ballot sheets by revealing the letter associated with Smith on the top ballot sheet, and the placement of that letter on the bottom sheet. We can do this on a scratch-off card with two rows of scratch-off pads. The first row will have a pad for each of the possible letters associated with Smith, the second row will have a pad for each of the possible positions. Thus, for a typical Punchscan ballot question with four candidates, our scratch off card would have a row of four pads labeled $a - d$ and a second row of pads labeled $1 - 4$.

The card needs to provide the voter with a challenge after the commitments are done. One way to do this is to have random integers under each pad. The voter would scratch off the pads associated with his top and bottom sheets, revealing two integers. The resulting sum of these integers would provide the challenge; an even sum would indicate a challenge for the top sheet, an odd sum a challenge for the bottom.

The attack works on the principle that knowledge of the top and bottom sheets, along with placement of the mark on the ballot, is sufficient to determine the cast vote. The scratch-off card contains commitments for the top and bottom sheets, with the receipt showing the voter's mark. Voters who fill out the scratch-off cards honestly (that is, in a manner consistent with their ballots) are forced to vote for the pledged candidate otherwise their deception would be detected. A voter attempting deception would have to incorrectly fill out the scratch-off card by misrepresenting the top or bottom sheet on the card. In that case, the voter's deception would be caught if the card's challenge asks for the misrepresented sheet.

Spoiling Ballots. The pledges on the scratch-off card pledge are meant to commit the voter to a particular ballot, but this commitment is weak. A determined voter might try to "cheat" the vote buying protocol by repeatedly spoiling ballots. For instance, he might vote for the wrong candidate and lie about one of the ballot sheets on the scratch off card with a 50% of being caught. If he will get caught, he could spoil ballots until obtaining one that will let him cheat.

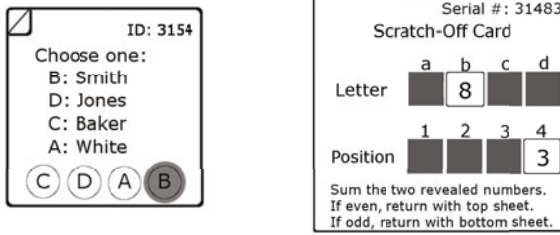


Fig. 6. A Committed Scratch-Off Card

However, introducing spoiled ballots allows us to create a more flexible attack by modifying the previous protocol. In this case, voters will strongly commit to a ballot and then find out whether or not to spoil that ballot.

Spoiled ballots play an important role in many paper-based E2E systems. Punchscan and Prêt-à-voter rely on the blank ballots being properly constructed, as was discussed in Section 3. One way to give voters assurance of proper construction is to let them audit ballots on election day. This would involve posting information about the audited ballot on a bulletin board and potentially allow the voter to leave the polling place with a blank ballot. If someone were to vote on the audited ballot (which is not allowed), the auditing information posted would let anyone see how that person voted. That is what will make this an effective attack.

The basic attack is an extension of the previous attack. In the case of Punchscan, voters must still commit to both ballot sheets. They must also commit to their ballot serial numbers, in order to prevent voters from repeatedly spoiling ballots. Thus, the scratch-off card would have three rows of scratch-off pads. The first two rows would be for the letter associated with the desired candidate on the top sheet and the location of that letter on the bottom sheet. The third row would be for the last digit of the ballot serial number and would have pads for the digits 0 – 9, perhaps with individual pads for multiple digits. As before, a random integer is underneath each pad. After scratching off the pads associated with his ballot, each voter would sum the three revealed integers. A sum congruent to 1 (mod 10) indicates the voter must spoil the current ballot and obtain a new ballot. Otherwise, the voter must cast the ballot. Either way, the voter returns to the buyer after voting and provides the scratch-off card and either the ballot receipt or the spoiled ballot. Figure 7 shows an example of a scratch-off card.

In this case, ballot spoiling serves to check that the voter filled out the scratch-off card correctly. A spoiled ballot allows the buyer to compare the scratch-off marks with the actual ballot, or some representation of that ballot on the bulletin board. Attempts at deception would be caught with 10% probability, which should be enough to voters. If the ballot is cast instead of spoiled, the buyer may assume the card was filled out correctly, and determine if the voter chose the proper candidate. For example, if the top sheet is returned, the buyer

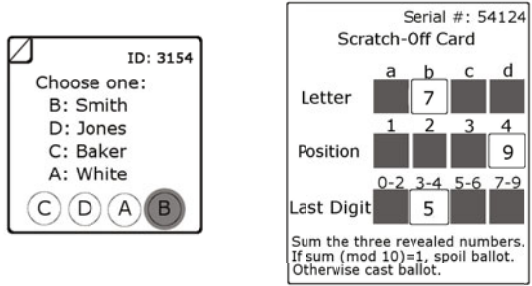


Fig. 7. A Committed Scratch-Off Card indicating spoil

can check that the candidate-letter mapping was pledged correctly, and that the location of the mark on the receipt matches the pledged location of the candidate letter on the scratch-off card.

While this variant of the scratch-off attack, in general, detects cheating with a lower probability than the previous version, it is more flexible. Namely, this variant no longer requires the voter to choose one of two possible receipts after filling out the card. Thus, it can be used against Punchscan even when procedures force voters to choose a receipt sheet prior to viewing the ballot, a successful countermeasure against the previous attack. Furthermore, it is effective against other paper-based E2E systems, like Prêt-à-voter . In that case, voters would be given scratch-off cards that allow them to commit to the cryptographic onion on each ballot, and the placement of the desired candidate on the left-hand ballot sheet.

4.4 Beacons

The scratch-off card attacks are effective because voters must first mark their ballots a particular way then learn from a challenge whether they need to perform an action that could reveal attempts at deception. More generally, we just need a communications channel between the seller and buyer after a ballot is marked. This channel could be as simple as the buyer holding up a small sign to voters as they are about to cast their marked ballots.

Alternatively we could use a chain of coerced voters that would vote in succession. Each voter would deliver a challenge to the preceding voter. To illustrate this attack, consider a ThreeBallot election. Each coerced voter would be instructed to fill out a hand-marked multiballot as shown in Figure 8. Voters would enter the poll booths in a chain, with the buyer sending in the next voter after the preceding voter has marked their ballot. The buyer would give the voter a challenge to pass on that instructs the previous voter to return with either the left, middle or right ballot. Voters who return with the correct receipt are rewarded. Furthermore, the challenging voters are rewarded if the challenge recipients return with the correct receipt.

150423	892314	239782
Smith ●	Smith ●	Smith ○
Jones ●	Jones ○	Jones ○
Baker ○	Baker ●	Baker ○
White ○	White ○	White ●

Fig. 8. ThreeBallot Marking Instructions

5 Conclusion

Procedural changes to the voting schemes can prevent most of the attacks discussed in this paper. Many of these attacks relied on the voter being free to make a choice after viewing the ballot that would determine what information is brought back from the poll booth; e.g., which receipt to take home or whether to cast or audit a ballot. Schemes which give voters that choice are vulnerable to coercion and vote buying attacks. However, procedural defenses can create additional vulnerabilities. For instance, if election officials ask voters if they will cast or audit a ballot prior to handing them one, the official could hand out misprinted ballots to those intended to cast the ballot. This decreases the risk of a vote buying attack, but increases the risk of election fraud, a serious attack.

End-to-end voting scheme designers should be wary to rely heavily on procedures to maintain their security properties. The advantages of end-to-end voting schemes over traditional systems are reduced when they rely on procedures; optical scan systems are relatively secure when proper chain of custody is maintained with the ballots. Simple attacks, like the misprinting attack described in this paper, can target these procedures to commit election fraud or violate privacy by changing the election procedures in seemingly inconsequential ways. The chances that a procedural change will go unnoticed increases as the number of procedural controls increases. In many instances, such changes will look like simple mistakes or oversights, rather than attempts at election fraud. While it is unrealistic to imagine schemes where specific procedures need not be followed to achieve security claims, a reasonable goal is to design systems whose verifiability claims are not dependent on the actions of election administrators or third-party auditors.

The field of end-to-end cryptographic voting schemes is still relatively young. Advances in the field of cryptography such as commitment schemes, signatures, secret sharing schemes and verifiable shuffles give us a variety of tools, but there is still room to improve the protocols which use these tools and the procedures that should be followed to mitigate threats.

Acknowledgments. The authors would like to thank to Ben Adida, Bill Burr, Richard Carback, Morris Dworkin, Ben Hosp, Andy Neff, Rene Peralta, Stefan Popoveniuc, Ron Rivest, Peter Ryan, Bruce Schneier, Poorvi Vora, and David Wagner for helpful discussions and comments on the topic of this paper.

References

1. Norden, L.: The machinery of democracy: Protecting elections in an electronic world. Technical report, Brennan Center Task Force on Voting System Security (October 2006), http://www.brennancenter.org/dynamic/subpages/download_file_38150.pdf
2. NIST: Threats to voting systems workshop (2005), <http://vote.nist.gov/threats/>
3. California Secretary of State: Top-to-bottom review (2007), http://www.sos.ca.gov/elections/elections/elections_vs.htm
4. Neff, C.A.: Practical high certainty intent verification for encrypted votes. VoteHere (2004), <http://www.votehere.net/vhti/documentation>
5. Adida, B., Neff, C.A.: Ballot casting assurance. In: EVT 2006: Proceedings of the USENIX/Accurate Electronic Voting Technology Workshop 2006 on Electronic Voting Technology Workshop, Berkeley, CA, USA, p. 7. USENIX Association (2006)
6. Popoveniuc, S., Hosp, B.: An introduction to Punchscan. In: Proceedings of Workshop on Trustworthy Elections (WOTE) (2006)
7. Fisher, K., Carback, R., Sherman, A.: Punchscan: Introduction and system definition of a high-integrity election system. In: Proceedings of Workshop on Trustworthy Elections (WOTE) (2006)
8. Rivest, R.: The ThreeBallot voting system. MIT (2006), <http://theory.lcs.mit.edu/~rivest/Rivest-TheThreeBallotVotingSystem.pdf>
9. Adida, B., Rivest, R.L.: Scratch & Vote: Self-contained paper-based cryptographic voting. In: WPES 2006: Proceedings of the 5th ACM workshop on Privacy in electronic society, pp. 29–40. ACM Press, New York (2006)
10. Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM* 24(2), 84–90 (1981)
11. Cohen, J.D., Fischer, M.J.: A robust and verifiable cryptographically secure election scheme. In: Proc. 26th IEEE Symp. on Foundations of Comp. Science, Portland, pp. 372–382. IEEE, Los Alamitos (1985)
12. Fujioka, A., Okamoto, T., Ohta, K.: A practical secret voting scheme for large scale elections. In: Zheng, Y., Seberry, J. (eds.) AUSCRYPT 1992. LNCS, vol. 718, pp. 244–251. Springer, Heidelberg (1993)
13. The Center for Responsive Politics: Opensecrets, <http://www.opensecrets.org>
14. Kelsey, J.: Strategies for software attacks on voting machines. In: Threats to Voting Systems Workshop (2005), http://vote.nist.gov/threats/papers/stategies_for_software_attacks.pdf
15. Chaum, D., Ryan, P.Y.A., Schneider, S.: A practical voter-verifiable election scheme. In: di Vimercati, S.d.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 118–139. Springer, Heidelberg (2005)
16. Rivest, R., Smith, W.: Three voting protocols: ThreeBallot, VAV and Twin. In: Proceedings of USENIX/ACCURATE Electronic Voting Technology Workshop (EVT) (2007)
17. Strauss, C.: The trouble with triples: A critical review of the triple ballot (3ballot) scheme. part 1. Verified Voting New Mexico (2006), <http://www.cs.princeton.edu/~appel/voting/Strauss-TroubleWithTriples.pdf>

18. Strauss, C.: A critical review of the triple ballot voting system. part 2: Cracking the triple ballot encryption. draft version 1.5. Verified Voting New Mexico (2006) <http://www.cs.princeton.edu/~appel/voting/Strauss-ThreeBallotCritique2v1.5.pdf>
19. Clark, J., Essex, A., Adams, C.: On the security of ballot receipts in e2e voting systems. In: Proceedings of Workshop on Trustworthy Elections (WOTE) (2007)
20. Moran, T., Naor, M.: Split-ballot voting: Everlasting privacy with distributed trust. In: Proceedings of Workshop on Trustworthy Elections (WOTE) (2007)
21. Essex, A., Clark, J., Carback, R., Popoveniuc, S.: The Punchscan voting system: Vocomp competition submission (2007), <http://www.punchscan.org/vocomp/PunchscanVocompSubmission.pdf>
22. Moran, T., Naor, M.: Polling with physical envelopes: A rigorous analysis of a human-centric protocol. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 88–108. Springer, Heidelberg (2006)

Aperio: High Integrity Elections for Developing Countries

Aleks Essex¹, Jeremy Clark², and Carlisle Adams¹

¹ University of Ottawa
aesse083@site.uottawa.ca,
cadams@site.uottawa.ca
² University of Waterloo
j5clark@cs.uwaterloo.ca

Abstract. This article presents an ‘end-to-end’ integrity verification mechanism for use in minimally equipped secret paper-ballot election environments. The scheme presented in this paper achieves high integrity properties without interfering with the traditional marking and tabulation procedures of paper-ballot elections. Officials and auditors can respectively generate and independently verify ‘end-to-end’ audit trails, with office stationery and entirely without cryptographic or mathematic computations.

1 Introduction

Aperio

-*verb* (Latin)

1. to reveal, uncover, lay bare.

Practical proposals for “end-to-end” election verification mechanisms have received much attention recently for their provision of election integrity *independent* of the physical chain-of-custody of ballots, achievable in part through the issuance of privacy-preserving receipts to voters [4,5,6,10]. These systems have been primarily cryptographic and largely inspired by the concept of anonymizing mixnets [3]. They also generally rely heavily on technology at the polling place and specialized technical knowledge for the verification process. These systems find their application predominantly in election environments with a pre-existing infrastructure of electronic (*e.g.*, optical scan) equipment. However the election environments which arguably would benefit the most from the end-to-end integrity properties—in developing democracies—are ones in which a technological infrastructure for voting is either not present, or not practical.

Three proposals were made in [14] for systems that do not directly utilize cryptography. However the first two proposals, ThreeBallot and VAV require the voter to mark the ballot in an arguably unintuitive way and still propose the use of some electronic equipment (to validate the ballot). The third and more elegant proposal, Twin, requires no electronic equipment or special ballot marking or

tabulation procedures, although does carry an inherent custody assumption that the “floating receipt” being issued to the voter is a *valid copy* of the ballot of another (anonymous) voter. In response, we propose Aperio, a simple paper-based election integrity verification mechanism for minimally equipped election environments, similar in intent to Twin, but with end-to-end integrity properties that do not rely on chain-of-custody.

2 Preliminaries

2.1 Integrity Properties

“*End-to-end*” (E2E) verification, as initially proposed in [7] offers two positive guarantees of integrity to voters:

1. Their ballot is included unmodified in the *same* set of ballots that get tallied,
2. The tally produced from this set is correct.

A less stringent requirement, “*software independence*” (SI), has also been proposed: “A voting system is software-independent if an undetected change or error in its software cannot cause an undetectable change or error in the election outcome” [13]. By definition, this term is not applicable to environments which do not utilize software components. However the problem has a physical analog to the so-called “chain of custody” of ballots. For our purposes we will defined a new term “*custody independence*” (CI) to be defined as: “A voting system is custody independent if an undetected change or error in its physical paper trail cannot cause an undetectable change or error in the election outcome.”

In informal terms, this requirement states that any entity that can manage to exchange the legitimate ballot box for another (rigged) one, cannot do so undetectably. Therefore any system that is end-to-end verifiable can be shown to exhibit the CI property. In the case of Twin [14], the first end-to-end criterion is not directly satisfied by the floating receipt model it employs.

2.2 Roles

In the following section we roughly define four sets of roles within the election, for which its members are not necessarily distinct (*e.g.*, a voter can also function as a verifier or poll official).

- **Election Trustees.** In addition to the existing requirements for the administration of a paper-ballot election, trustees oversee the generation, commitment and decommitment of the audit commitment lists of the corresponding ballots.
- **Verifiers.** An (unspecified) partnership of concerned entities, including potentially candidates and their representatives, non-governmental organizations, voter advocacy groups and other election observers act to verify the integrity (*i.e.*, correctness) of the election outcome *independently* of physical access to the official paper trail via the mechanism presented in the following section.

- **Voters.** Those who vote in the election. In addition to voting, voters are given the option of retaining a (privacy-preserving) receipt.
- **Poll Officials.** Those responsible for administering the election at the polling location with tasks that include voter authentication, ballot distribution and casting.

3 Basic Paper Scheme

3.1 Ballot Format

In a conventional paper-ballot election, a voter is issued a single sheet of paper—a ballot. Under the Aperio scheme, a voter is instead issued a “ballot assembly” which consists of a paper ballot sheet, a receipt sheet and any number of audit sheets stacked and joined in such a way that only the top ballot layer is visible to the voter. These layers are defined as follows:

- A “*ballot*” is used to describe any paper *Australian ballot*¹ with the specific property that the list of candidates or proposals is printed in an independently random order across the set of ballots in an election,
- a “*receipt*” is used to describe a sheet of paper, equivalent in size and layout to a “ballot,” but without a candidate list, and additionally a unique serial number, and
- an “*audit sheet*” is used to describe a sheet of paper, equivalent in size and layout to a “ballot” but without a candidate list. It does not contain a serial number, but has pre-printed regions in which to write one. Additionally provided is a region in which to mark a “commitment reference number.”

For simplicity of the following description, we will consider the base-case ballot assembly which includes *two* audit sheets. We will refer to the assembly’s layers by a standard office-paper color pallet, in which the ballot, receipt, and two audit sheets are assigned the colors “white,” “canary,” “goldenrod,” and “pink” respectively (see Figure 1). The sheets are stacked in such a way that voter-made marks on the ballot sheet will be transferred to the other sheets using carbon-copy, or alternatively NCR brand (carbonless copy) paper (see Figure 2).

3.2 Initial Setup

As in other of E2E systems [4,5,6,10], the entity responsible for printing ballots is generally entrusted with voter privacy. Although there likely exist protocols under which ballots could be printed in either a threshold-trust or even a fully oblivious manner, for simplicity we describe the following operations as being performed by a single privacy-entrusted entity.

¹ “An official ballot printed at public expense on which the names of all the candidates and proposals appear and which is distributed only at the polling place and marked in secret” [2].

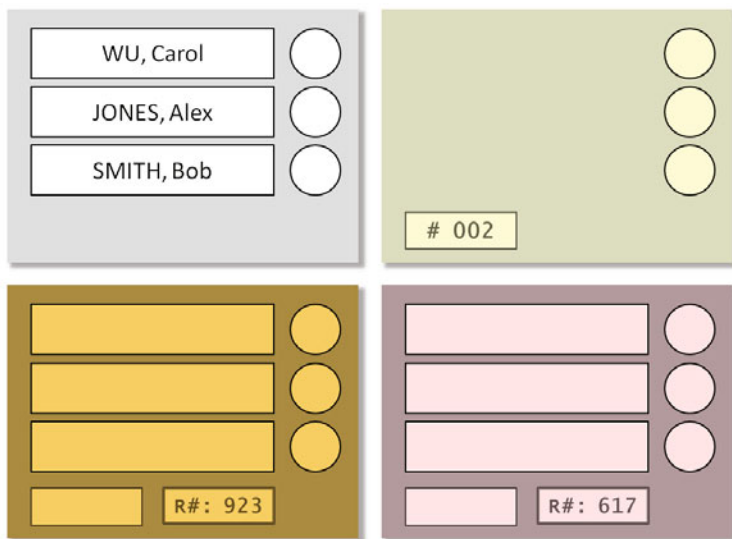


Fig. 1. Ballot Assembly: (Top left) Paper “Australian” ballot with random candidate order (*white*). (Top right) Receipt with unique serial number (*canary*). (Bottom left) Audit sheet with “commitment reference number” (*goldenrod*). (Bottom right) Audit sheet with independently assigned “commitment reference number” (*pink*).

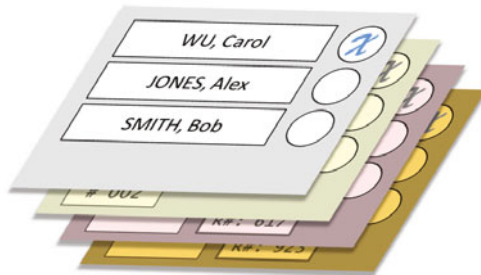


Fig. 2. Ballot Assembly (Exploded View): Marks made on the top “ballot” layer are transferred to the “receipt” and “audit sheet” layers using “carbon-copy” style paper

Generating Ballots. In order to preserve voter privacy, the association of candidate order and serial number must be secret and arbitrary, such that knowledge of one in no way implies knowledge of the other. Consider a stack of b ballots, each with an independently randomly printed candidate order. Likewise consider a stack of b receipts each with unique serial number. An arbitrary association can be formed by drawing the top ballot and receipt sheets from the respective stacks and joining (*e.g.*, stapling) them together. Additionally the (blank) audit sheets are joined to the ballot and receipt sheets, constituting a specific instance of a ballot assembly. This is repeated to create b independent ballot assemblies.

Generating Commitment Lists. A “commitment list” is defined as a list of b rows pre-printed with a monotonically increasing set of b “commitment reference numbers.” Next to each commitment reference number is a region, initially blank, that will contain an associated value. We define two types of commitment lists:

- **Receipt commitment lists** contain a set of b distinct commitment reference numbers, each with a (randomly) associated *serial number*
- **Ballot commitment lists** contain the same set of b distinct commitment reference numbers, each with a (randomly) associated *candidate list ordering*.

To generate the commitment lists, we begin by considering a particular audit trail color (e.g., pink). The *pink receipt commitment list* and *pink ballot commitment list* are generated as follows:

1. A ballot assembly is drawn from the stack and the serial number s is noted. A non-replaced random number $i \in b$ is selected (e.g., on a slip of paper drawn from a hat),
2. The pink audit sheet of the ballot is exposed, and the number i is written in the commitment reference number space,
3. On the *pink receipt commitment list*, the number s is written in the blank space beside commitment reference number i ,
4. On the *pink ballot commitment list*, the candidate order o is written in the blank space beside commitment reference number i .

These steps are performed on all ballot assemblies. The pink audit trail is now complete and the *goldenrod receipt commitment list* and *goldenrod ballot commitment list* (and any additional audit trail colors) can be generated in the same manner. An example ballot assembly and corresponding entries in the commitment lists is depicted in Figure 3.

Committing. For an election with two audit trails (pink and goldenrod), the election trustees generate the *pink receipt*, *pink ballot*, *goldenrod receipt* and *goldenrod ballot commitment* lists. They lock these values in time (i.e., commit to them) through the following procedure:

1. Each commitment list is placed in its own appropriately labeled tamper-evident document envelope and sealed,
2. The trustees present the sealed envelopes to the verifiers who are given the opportunity to inspect the exterior of the envelope and sign on the flaps,
3. The envelopes are returned into the custody of the trustees.

3.3 Print Audit Selections

In order to ensure the commitment reference numbers of the ballot assemblies point to the same candidate orderings/serial numbers appearing on the ballot and receipt layers, some ballot assemblies will be selected by the verifiers for a “print audit.” Procedure could vary between jurisdictions, but one recommendation would be for the print audit selections to be made in conjunction with

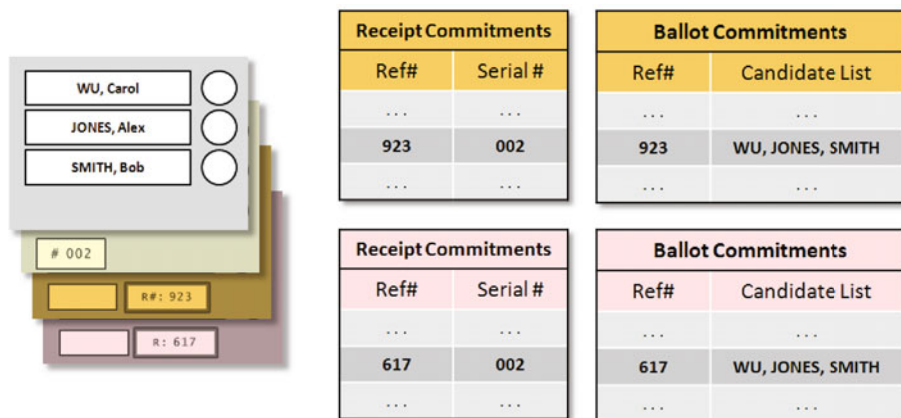


Fig. 3. Commitment Lists: For *each* audit trail (goldenrod and pink in this case) two commitment lists are generated – a **receipt** list and a **ballot** list, each randomly associating serial numbers and respectively candidate orderings with a distinct commitment reference number

the random spot checks of the poll registration book at the polling place during election day (as is the procedure in many paper ballot elections today). A verifier would select a ballot at random from the stack of ballots, and the poll worker would mark the ballot as spoiled (*e.g.*, by punching a hole through the layers). The spoiled ballot would then be given to the verifier to retain for a later auditing procedure.

3.4 Voting

Voting is conducted in accordance with the jurisdiction’s procedures for paper ballot elections. An eligible and authenticated voter is issued a ballot assembly by a poll official and is directed to a voting booth. The voter marks the ballot assembly as they normally would in any conventional paper ballot election. They return the ballot assembly to the poll official who first inspects the assembly to ensure the respective layers are still sealed together. The official then separates and distributes the ballot layers in the following way: the ballot layer is cast in the ballot box. The receipt (canary) layer is issued to the voter as a receipt. The pink and goldenrod audit sheets are cast into “pink” and “goldenrod” audit boxes respectively.

3.5 Election Outcome

After the close of the polls, the election results are tallied normally, in accordance with the pre-existing procedures of the jurisdiction for paper ballot elections using contents of the ballot box and is referred to as the “official tally.” At the close of the polls, the “pink” and “goldenrod” audit boxes are relinquished into custody of the verifiers.

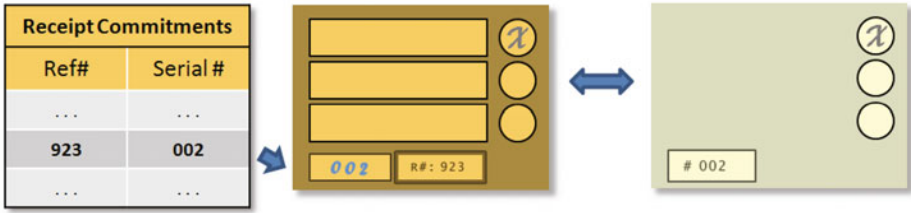


Fig. 4. Reconstituting the Receipt Audit Trail: The Decommitted (goldenrod) receipt commitment list (**Left**) can be used to reconstitute receipts from the corresponding (goldenrod) audit trail (**Middle**). The reconstituted receipt audit trail can be cross-referenced against voter receipts (**Right**).

3.6 Decommitting

A coin is flipped in public. If the outcome is heads, the pink audit box is selected to become the ballot audit trail and the goldenrod audit box is selected to become the receipt audit trail. If the outcome is tails, the opposite envelopes are selected. Trustees respond by releasing (*i.e.*, decommitting) the corresponding commitment envelopes through the following procedure:

1. Trustees relinquish selected commitment envelopes into the custody of the verifiers,
2. Verifiers inspect envelopes for the presence of their signatures on the flap and for the absence of evidence of physical tampering of the envelope,
3. Trustees destroy (*e.g.*, shred) the remaining unselected commitment envelopes.

In the following explanation of the receipt and tally audit, for clarity, we will assume a case in which *heads* was the outcome of the coin flip—meaning goldenrod and pink were selected to become the receipt and ballot audit trails respectively.

3.7 Receipt Audit

Using the contents of the goldenrod audit trail box in conjunction with the goldenrod receipt commitment list, a receipt trail can be reconstituted in the following way (see Figure 4):

1. A goldenrod audit sheet is drawn from the goldenrod audit trail box. The commitment reference number $i \in b$ is noted,
2. The i -th row of the *goldenrod receipt commitment list* is consulted, and corresponding serial number s is noted,
3. The number s is written into the blank serial number space on the audit sheet.

These steps are performed on all goldenrod audit sheets. The reconstituted receipts can be cross-referenced against voter receipts to ensure the records match.

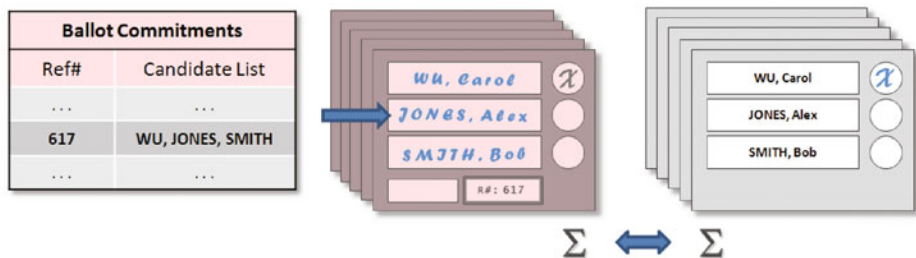


Fig. 5. Reconstituting the Ballot Audit Trail: The Decommited (pink) ballot commitment list (Left) can be used to reconstitute ballots from the corresponding (pink) audit trail (Middle). The reconstituted ballot audit trail can be tallied and the totals cross-referenced against the official tally (Right).

Voters could optionally give their receipts to the verifiers to cross-reference on their behalf, or alternatively the verifiers could publish the reconstituted receipt trail in a public venue (e.g., newspaper) with which voters could check for themselves.

3.8 Tally Audit

Using the contents of the pink audit trail box in conjunction with the pink ballot commitment list, a ballot can be reconstituted in the following way (see Figure 5):

1. A pink audit sheet is drawn from the pink audit trail box. The commitment reference number $j \in b$ is noted,
2. The j -th row of the *pink ballot commitment list* is consulted, and corresponding candidate list ordering o noted,
3. The candidates are written in order o into the blank candidate name spaces on the audit sheet.

These steps are performed on all pink audit sheets. The reconstituted ballots can be tallied and cross-referenced against the official tally to ensure a match.

3.9 Print Audit

Recalling the randomly selected (spoiled) ballots from section 3.3, the correctness of a ballot assembly’s printing can be verified in the following way:

1. For a given ballot assembly, candidate order o and serial number s are noted,
2. The ballot assembly’s pink and goldenrod audit layers are reconstituted into ballot and receipt audit layers as described in sections 3.7 and 3.8 to recover the o' and s' pointed to by the respective commitment reference numbers,
3. The printing of this given ballot assembly is correct if $o = o'$ and $s = s'$.

4 Security Analysis

In this section, we describe how Aperio meets the E2E integrity criteria as defined in section 2. Further, we analyse the attack vectors that an adversary could use to attempt to corrupt the results of an election and demonstrate the protections offered by Aperio to thwart these attacks.

4.1 A Positive Assertion of Security

Let an unmarked Aperio ballot assembly be the tuple $\langle o, s, c_p, c_g \rangle$ for candidate order, serial number, commitment reference number of the pink sheet, and commitment reference number of the goldenrod sheet. Let ρ denote the position marked by the voter on each element of the ballot assembly. For the following discussion, again consider the instance in which the *pink receipt commitment list* and the *goldenrod ballot commitment list* were selected to be decommitted (although the following security properties are invariant to any particular selection). The audit process establishes the following facts:

1. The voter's receipt contains $\langle s, \rho \rangle$. By matching the voter's receipt to the *receipt commitment list*, it can be verified that ρ' (of row s of the receipt commitment list) matches the ρ on the voter's receipt. Therefore, the voter's mark is included unmodified in the collection of ballots—the first E2E criterion.
2. The print audit verifies that s and c_p printed on a ballot are the same as in the commitment reference sheet and additionally,
3. Verifies o and c_g are the same as on the ballot reference sheet.
4. Since 2 and 3 are dependent on a random decision, it is probabilistic that s and c_g also are consistent between the printed ballots and reference sheets, and additionally,
5. It is probabilistic that o and c_p also are consistent between the printed ballots and reference sheets. If the printed ballots are not consistent, this would be detected with probability $1 - (1 - Y)^{x-1}$ where Y is the percentage of receipts checked and x is the number of audit sheets.
6. By combining facts 2 and 5, or similarly 3 and 4, we infer that s and o on the sheets are consistent with what the voter saw in the polling booth.
7. By combining 1 and 6, the voter is assured that the same ρ at s on their receipt is in the ballot commitment list somewhere beside the same o that was on their ballot.
8. Finally, given 7, the voter can generate a correct tally for all votes using the ballot reference sheet proving that the collection of ballots is tallied correctly—the second property of an E2E election.

The indirectness of this proof prevents the voter from proving which candidate they voted for to a coercer or someone wishing to purchase their vote. The tally that was generated to provide fact 8 can also be compared to the official tally generated using the original paper ballots for additional assurance.

4.2 Prevented Attacks

In addition to the positive security properties already outlined, it may be also useful to demonstrate a set of attacks it is not susceptible to.

- **Adding or removing ballots.** In order to increase the number of votes for a candidate, an adversary may “stuff” the ballot box with extra ballots for their candidate of choice. Alternatively, given that elections are local events and that correlations often exist between a locality and a political preference, an adversary may attempt to destroy cast ballots in a discriminatory manner by choosing locations based on expected political preference. We assume that a voter registration list is maintained with the number of voters who cast ballots in the election. This information may also be corroborated by election observers. The number of ballots and audit sheets should be identical between them and should also be identical to the total voters on the registration.
- **Modifying ballots.** A more sophisticated adversary would avoid upsetting the number of cast ballots by either replacing existing ballots with new ballots marked for their candidate or modifying the marks on existing ballots. Since a ballot assembly is distributed between distinct boxes, the tuple $\langle o, s, c_p, c_g \rangle$ becomes unlinked and subsequently unassociable once cast into the respective boxes. If an adversary modifies or replaces o on a ballot, then the audit tally will not match the tally generated from adding up all the top layers. Its not in the interest of an adversary to modify c_p or c_g , as there is no way of knowing which candidate a vote is for and which it will be mapped to if modified. Furthermore, there is only a 50% chance that the tally will change at all—alternatively the box will be used to decommit receipts. In either case, such modifications will be detectable either because the tallies will not match, or the receipts will not match. Since c_g and c_p hide o , the voter cannot change both an o on a ballot and, say, c_g on a goldenrod audit sheet in a way that consistently changes both tallies, should the goldenrod ballot commitment list be opened.
- **Misprinting ballots.** An adversary could also misprint ballot assemblies before voting day, or mix-and-match sheets from one ballot assembly with sheets of another. Most of these attacks will not affect the tally (only the receipts) and there is only one method for potentially modifying the tally in an undetectable way: switch both o and one audit sheet, say c_g , with another ballot with a different order o' , where $o \neq o'$. If the *goldenrod ballot commitment list* is selected to be decommitted, the two tallies will match, otherwise the attack will be detected. Furthermore, since the adversary has no guarantee of which voter will receive the misprinted ballot, votes cannot be predictably directed to a particular favored candidate. This attack is marginal and can be further mitigated by using more than two audit layers, which exponentially decreases the probability of a successful execution of this attack.
- **Forced randomization attack.** An adversary could coerce a voter into returning with a receipt with a particular position marked. Since the adversary

has no knowledge of o , this is equivalent to forcing the voter to throw away their vote by voting for a random candidate as demonstrated in [12]. Since a random vote will be given to each candidate with equal probability, coercing a random votes will have the same effect as coercing a voter to not vote at all—the latter likely being easier to execute.

- **Chain voting.** An adversary who can capture an unmarked ballot can execute a chain voting attack where they fill out the unmarked ballot for a candidate they choose, then coerce a voter into casting the adversary’s ballot in place of the unmarked ballot the voter receives, and demand that the voter return to them the voter’s unmarked ballot so they can execute the attack *ad infinitum*. This can be mitigated by the use of a detachable, uniquely marked “counterfoil.” As the voter registration official issues the ballot, the number n on the counterfoil is noted. When the voter returns, the official verifies the presence of the counterfoil and notes the number n' . If $n = n'$, the voter has not exchanged ballots. The counterfoil is then detached and discarded, and the ballot assembly and the casting procedure continues as in section 3.4. As an option, the counterfoil could be additionally utilized to initially seal the ballot layers together to prevent anyone from viewing s , c_g and c_p beforehand.
- **Pattern Attacks.** A voter wishing to sell their vote could mark their ballot in a pattern that is likely to be uniquely identifiable and then point out its location to the buyer in the ballot commitment list. This can be mitigated by partitioning the audit into subaudits: *i.e.*, having separate commitment lists for each contest or small collections of contests. Exactly how to partition a ballot to marginalize a pattern attack can be determined statistically—*i.e.*, [8].

5 Privacy

In this section we describe how the scheme protects voter privacy. With the exception of the privacy entrusted entity that generates and prints ballot assemblies, no information about how a voter votes becomes known to the other entities (*i.e.*, voter, poll officials, verifiers) with the following justification:

- **Ballot assembly.** It is easy to see that through the decommitting process, $\langle c_p, c_g \rangle \Rightarrow \langle o, s \rangle$, and therefore $\langle c_p, c_g, \rho \rangle \Rightarrow \langle o, s, \rho \rangle$, which is the basis of the print audit of *spoiled* ballot assemblies. Privacy is preserved on these ballot assemblies given $\rho = \emptyset$. Assuming poll procedure is followed, then $\langle c_p, c_g \rangle$ will be physically unlinked by the poll official and logically unlinked by the mixing in the audit trail ballot boxes. It is central to voter privacy that neither the voters, poll officials or verifiers see $\langle c_p, c_g \rangle$ leading up to, and during, the ballot casting process. The assembly layers might be sealed (*e.g.*, glued) together as previously suggested by a tear-off “counterfoil.”
- **Ballot receipt.** Given that $\langle o, s \rangle$ were randomly selected in section 3.2, and known only to the privacy entrusted entity, then to all other entities $\langle s \rangle \not\Rightarrow \langle o \rangle$ and therefore $\langle s, \rho \rangle \not\Rightarrow \langle o, s, \rho \rangle$ given receipt $\langle s, \rho \rangle$.

- **Receipt Commitment Lists.** Given *ballot commitment lists* $\mathcal{CB}_g = \{\langle c_{g_1}, o_1 \rangle, \dots, \langle c_{g_b}, o_b \rangle\}$ and $\mathcal{CB}_p = \{\langle c_{p_1}, o_1 \rangle, \dots, \langle c_{p_b}, o_b \rangle\}$ and *receipt commitment lists* $\mathcal{CR}_g = \{\langle c_{g_1}, s_1 \rangle, \dots, \langle c_{g_b}, s_b \rangle\}$ and $\mathcal{CR}_p = \{\langle c_{p_1}, s_1 \rangle, \dots, \langle c_{p_b}, s_b \rangle\}$, but given that only one of $\langle \mathcal{CB}_g, \mathcal{CR}_p \rangle$ and $\langle \mathcal{CR}_g, \mathcal{CB}_p \rangle$ are ever made public, it is easy to see $\langle c_{g_i}, o_i \rangle \not\Rightarrow \langle c_{g_i}, s_i \rangle$ and $\langle c_{p_i}, o_i \rangle \not\Rightarrow \langle c_{p_i}, s_i \rangle$ and thus $\langle c_{g_i}, \rho_i \rangle \not\Rightarrow \langle o_i, s_i, \rho_i \rangle$ and likewise $\langle c_{p_i}, \rho_i \rangle \not\Rightarrow \langle o_i, s_i, \rho_i \rangle$.

5.1 Prevented Attacks

Under the privacy properties, a link cannot be established between a vote and a receipt, and therefore the receipt cannot be used to prove how a voter voted. That is to say any observer, given only the ballot receipt, cannot “guess” a voter’s selections with non-negligible advantage. These privacy properties address the following attacks:

- **Vote Buying.** A voter who votes a certain way, following an arrangement made between the voter and any entity *a priori* to voting, cannot subsequently prove how they voted. This effectively relegates vote buying to, at best, conventional threats, or at worst, to the previously mentioned forced-randomization attack.
- **Retribution.** A subtly different threat, often overlooked in literature, is the circumstance in which no precursory voting agreement exists, yet a possibility of retribution exists if the voter’s selections become known *a posteriori* to voting. Concern for future retribution would be legitimate cause for a voter to vote differently than intended, and therefore is arguably as serious a concern as vote buying and must likewise not be facilitated by any proposed receipt scheme.

6 Extensions and Other Applications

6.1 Multiple Contests

The specific system presented in this paper considered a small, single contest race. Under this scheme, uniquely patterned receipts are unlikely to occur under the so-called “Short Ballot Assumption” of [14]. In the case of cryptographic E2E systems, elections with multiple contests have attempted to mitigate against pattern attacks by partitioning the anonymizing network by contest [11]. It is easy to see the physical analog of this technique for a multi-contest ballot would be to either to perforate the ballot assembly such that the assembly could be separated by contest at casting time, or simply to have one assembly per contest (a technique already employed by some jurisdictions).

6.2 Automation of Verification Process

Although an election environment may not provide for optical scanners at the polling place, it may be reasonable to assume some computing capability on

behalf of the trustees and verifiers. In this circumstance, the trustees might print the commitment reference numbers on audit sheets as well as the commitment lists in an optically readable format to speed the verification process. In this scheme, the same audit operations would be undertaken, except by a computer.

6.3 Implications for Cryptographic Schemes

The anonymizing network model introduced by Aperio is fundamentally simpler than the mixnet model employed by many cryptographic schemes in the sense that it does not propagate mark states through mix nodes, and therefore does not directly require a scheme such as *randomized partial checking* [9] to verify the network's correctness. This could replace the anonymizing network of a system such as Scantegrity [4] which employs multiple instances of a two stage permutation-network, instead, with multiple (independently shuffled) instances of $(\mathcal{CR}, \mathcal{CB})$. This change would likely decrease the total number of verification operations, and arguably decreases the overall conceptual complexity of the verification process.

7 Conclusion

In this document we have introduced Aperio, a mechanism to provide custody independent verification of election results without the use of electronic equipment or special skill requirements for voting, tallying, or verification. Our intent through this proposal is twofold. First, we hope to cultivate discussion on practical techniques for strengthening the democratic process in developing counties and other minimally equipped or minimally funded election environments. Second, it is our hope that Aperio will be useful as a more broadly-accessible educational tool for E2E concepts, especially for the purpose of growing understanding and acceptance of its various cryptographic counterparts.

References

1. Adida, B., Rivest, R.L.: Scratch & vote: self-contained paper-based cryptographic voting. In: WPES 2006: Proceedings of the 5th ACM workshop on Privacy in electronic society, pp. 29–40 (2006)
2. Ballot, A.: Merriam-webster dictionary. Online
3. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM* 24(2), 84–90 (1981)
4. Chaum, D., Essex, A., Carback, R., Clark, J., Popoveniuc, S., Sherman, A.T., Vora, P.: Scantegrity: End-to-end voter verifiable optical-scan voting. *IEEE Security and Privacy Magazine* (May/June 2008)
5. Chaum, D., Ryan, P.Y.A., Schneider, S.A.: A Practical, Voter-verifiable, Election Scheme. Technical Report Series CS-TR-880, University of Newcastle Upon Tyne, School of Computer Science (December 2004)
6. Chaum, D., van de Graaf, J., Ryan, P.Y.A., Vora, P.L.: Secret ballot elections with unconditional integrity. Technical report, IACR Eprint (2007), <http://eprint.iacr.org/>

7. United States Election Assistance Commission. 2005 voluntary voting system guidelines (December 2005), http://eac.gov/vvsg_intro.htm
8. Henry, K., Stinson, D., Sui, J.: The effectiveness of receipt-based attacks on three-ballot. Technical report, Centre for Applied Cryptographic Research, University of Waterloo (2007)
9. Jakobsson, M., Juels, A., Rivest, R.L.: Making mix nets robust for electronic voting by randomized partial checking. In: USENIX Security 2002, pp. 339–353 (2002)
10. Popoveniuc, S., Hosp, B.: An Introduction to Punchscan. In: Preproceedings of the 2006 IAVoSS Workshop on Trustworthy Elections, Robinson College, Cambridge, United Kingdom. International Association for Voting System Sciences (2006)
11. Popoveniuc, S., Stanton, J.: Undervote and pattern voting: Vulnerability and a mitigation technique (June 2007)
12. Popoveniuc, S., Stanton, J.: Buying random votes is as hard as buying no-votes. Cryptology ePrint Archive, Report 2008/059 (2008), <http://eprint.iacr.org/>
13. Rivest, R.L., Wack, J.: On the notion of “software independence” in voting systems. DRAFT Version (July 28, 2006)
14. Rivest, R.L., Smith, W.D.: Three Voting Protocols: Threeballot, VAV, and Twin. In: Usenix/Accurate EVT (August 2007)

Author Index

- Adams, Carlisle 388
Araújo, Roberto 274, 330
Bruck, Shuki 97
Carback, Richard 357
Catalano, Dario 37
Chaum, David 357, 370
Chevallier-Mames, Benoît 191
Clark, Jeremy 357, 388
Custódio, Ricardo Felipe 274
Delaune, Stéphanie 289
Essex, Aleks 357, 388
Ferreira, Paulo 310
Fouille, Sébastien 330
Fouque, Pierre-Alain 191
Furukawa, Jun 141
Gerck, Ed 1
Goler, Jonathan A. 83
Graaf, Jeroen van de 231, 274
Hirt, Martin 64
Hosp, Ben 242
Imai, Hideki 107
Jakobsson, Markus 37
Jefferson, David 97
Joaquim, Rui 310
Jones, Douglas W. 175
Jonker, Hugo 216
Juels, Ari 37
Kelsey, John 370
Kiayias, Aggelos 155
Kremer, Steve 289
Kutyłowski, Mirosław 343
Lundin, David 260
Moran, Tal 370
Mori, Kengo 141
Otsuka, Akira 107
Peacock, Thea 200
Pieters, Wolter 216
Pointcheval, David 191
Popoveniuc, Stefan 242, 357
Regenscheid, Andrew 370
Rezende, Pedro A.D. 124
Ribeiro, Carlos 310
Rivest, Ronald L. 97
Ryan, Mark 289
Ryan, Peter Y.A. 200
Sako, Kazue 141
Selker, Edwin J. 83
Stern, Julien 191
Traoré, Jacques 191, 330
Yung, Moti 155
Zagórski, Filip 343