

Aggelos Kiayias
Helger Lipmaa (Eds.)

LNCS 7187

E-Voting and Identity

Third International Conference, VotelD 2011
Tallinn, Estonia, September 2011
Revised Selected Papers



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 for Informatics, Saarbruecken, Germany

Aggelos Kiayias Helger Lipmaa (Eds.)

E-Voting and Identity

Third International Conference, VoteID 2011
Tallinn, Estonia, September 28-30, 2011
Revised Selected Papers



Springer

Volume Editors

Aggelos Kiayias
National and Kapodistrian University of Athens
Department of Informatics and Telecommunications
Panepistimiopolis, 15784 Athens, Greece
E-mail: aggelos@di.uoa.gr

Helger Lipmaa
University of Tartu, Institute of Computer Science
J. Liivi 2, 50409 Tartu, Estonia
E-mail: helger.lipmaa@gmail.com

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-32746-9 e-ISBN 978-3-642-32747-6
DOI 10.1007/978-3-642-32747-6
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2012944270

CR Subject Classification (1998): E.3, D.4.6, K.6.5, C.2, J.1, K.4.4, K.5.2

LNCS Sublibrary: SL 4 – Security and Cryptology

© Springer-Verlag Berlin Heidelberg 2012

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.

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

These are the proceedings of VoteID 2011, the third in the series of International Conferences on E-Voting and Identity. The conference was held in Tallinn, Estonia, during September 28–30, 2011. The previous two VoteID conferences were held in 2007 (Bochum) and 2009 (Luxembourg). Since then, several countries have moved forward (and a few, backward) on e-voting. In particular, Estonian parliamentary e-voting in Spring of 2011 resulted both in a record number of remote votes (140,846 voters chose to vote remotely) and new controversies. In parallel, Norway has gone forward to start its own remote e-voting process. The current proceedings contain papers that describe both systems.

Since e-voting is already applied in the real world (and often, we have to say, not in an ideally secure way), research on e-voting is gaining in importance. We hope that VoteID 2011 helped not only to further academic research on e-voting, but also to tighten the contacts between the theory and the practice of the discipline. In particular, the Program Committee accepted works on both theoretical and practical aspects (“experience” or system-oriented papers). Moreover, the two invited talks, one by Henrik Nore and Ida Stenerud and the second one by Jens Groth, were explicitly chosen to reflect both sides of e-voting research. We would like to thank the invited speakers for accepting our invitation and for delivering excellent talks. In addition, VoteID 2011 has a panel on “Verifiable E-Voting and Real World,” and we would like to thank the panelists (Tarvi Martens, Kristian Gjøsteen, Henrik Nore, Alexander Trechsel, and Andrew Regenscheid) for participation. The Program Committee selected 15 papers for presentation at the conference out of a total of 33 anonymous submissions. Each submission was reviewed by at least three Program Committee members, while Program Committee submissions were reviewed by at least four Program Committee members.

We would like to thank everyone who helped make this conference happen. Our thanks go to the Program Committee and their subreviewers, as listed on the following pages. The submission and review process was greatly simplified by the Web submission and review software written by Shai Halevi. We thank all the submitters as well as the authors for revising their papers accordingly to the reviewers’ suggestions. The revised versions were not checked by the Program Committee so the authors bear full responsibility for their contents. We thank Nikolaos Karvelas of the University of Athens and the staff at Springer for their help with producing the proceedings.

VoteID 2011 was generously supported by Scytl. We note that these proceedings include a paper authored by Scytl employees. This contribution was evaluated by the Program Committee entirely independently of the company's support. In the local organization, the General Chair (H. Lipmaa) was helped very professionally by a team, lead by Kerli Kangro, from the conference center of the Tallinn University.

January 2012

Aggelos Kiayias
Helger Lipmaa

VoteID 2011

The Third International Conference on E-voting and Identity 2011

Tallinn, Estonia
September 28–30, 2011

Program Chairs

Aggelos Kiayias	University of Athens, Greece
Helger Lipmaa	University of Tartu, Estonia

Program Committee

Josh Benaloh	Microsoft Research, USA
Felix Brändt	Technische Universität München, Germany
Yvo Desmedt	University College London, UK
Edith Elkind	Nanyang Technological University, Singapore
Jens Groth	University College London, UK
Joseph Lorenzo Hall	UC Berkeley and Princeton, USA
Hugo Jonker	University of Luxembourg, Luxembourg
Aggelos Kiayias	University of Connecticut, USA
Helger Lipmaa	University of Tartu, Estonia
Tal Moran	Harvard, USA
Rene Peralta	NIST, USA
Olivier Pereira	Université Catholique de Louvain, Belgium
Mark Ryan	University of Birmingham, UK
Peter Ryan	University of Luxembourg, Luxembourg
Berry Schoenmakers	University of Eindhoven, The Netherlands
Jörg Schwenk	Ruhr-Universität Bochum, Germany
Vanessa Teague	University of Melbourne, Australia
Melanie Volkamer	TU Darmstadt, Germany
Moti Yung	Columbia University and Google, USA

External Reviewers

Haris Aziz
Catherine Baker
Stephanie Bayer
Markus Brill
Marco Cova
Chris Culnane
Denise Demirel
Naipeng Dong
Richard Frankland
Kristian Gjøsteen
Paul Harrenstein
Simon Kramer

Dalia Khader
Gabriele Lenzini
Maina Olembo
Doron Peled
Ariel Procaccia
Kim Ramchen
Michael Schläpfer
Hans Georg Seedig
Nicolas Troquard
Dominique Unruh

Table of Contents

Norwegian Internet Voting

The Norwegian Internet Voting Protocol	1
<i>Kristian Gjøsteen</i>	
Transparency and Technical Measures to Establish Trust in Norwegian Internet Voting	19
<i>Oliver Spycher, Melanie Volkamer, and Reto Koenig</i>	
Internet Voting System with Cast as Intended Verification	36
<i>Jordi Puiggalí Allepuz and Sandra Guasch Castelló</i>	

Voting Systems 1

Linear Logical Voting Protocols	53
<i>Henry DeYoung and Carsten Schürmann</i>	
Efficient Vote Authorization in Coercion-Resistant Internet Voting	71
<i>Michael Schlöpfer, Rolf Haenni, Reto Koenig, and Oliver Spycher</i>	
The Bug That Made Me President a Browser- and Web-Security Case Study on Helios Voting	89
<i>Mario Heiderich, Tilman Frosch, Marcus Niemietz, and Jörg Schwenk</i>	

Voting Systems 2

An Efficient and Highly Sound Voter Verification Technique and Its Implementation	104
<i>Rui Joaquim and Carlos Ribeiro</i>	
Single Layer Optical-Scan Voting with Fully Distributed Trust	122
<i>Aleksander Essex, Christian Henrich, and Urs Hengartner</i>	
Paperless Independently-Verifiable Voting	140
<i>David Chaum, Alex Florescu, Mridul Nandi, Stefan Popoveniuc, Jan Rubio, Poorvi L. Vora, and Filip Zagórski</i>	

Prêt à Voter and Trivitas

Feasibility Analysis of Prêt à Voter for German Federal Elections	158
<i>Denise Demirel, Maria Henning, Peter Y.A. Ryan, Steve Schneider, and Melanie Volkamer</i>	

Prêt à Voter with Write-Ins 174
*Steve Schneider, Sriramkrishnan Srinivasan, Chris Culnane,
James Heather, and Zhe Xia*

Trivitas: Voters Directly Verifying Votes 190
Sergiu Bursuc, Gurchetan S. Grewal, and Mark D. Ryan

Experiences

The Application of I-Voting for Estonian Parliamentary Elections
of 2011 208
Sven Heiberg, Peeter Laud, and Jan Willemson

Towards Best Practice for E-election Systems: Lessons from Trial and
Error in Australian Elections 224
Richard Buckland, Vanessa Teague, and Roland Wen

On the Side-Effects of Introducing E-Voting 242
James Heather, Morgan Llewellyn, Vanessa Teague, and Roland Wen

Author Index 257

The Norwegian Internet Voting Protocol

Kristian Gjøsteen*

Norwegian University of Science and Technology
kristian.gjosteen@math.ntnu.no

Abstract. The Norwegian government will run a trial of internet remote voting during the 2011 local government elections. A new cryptographic voting protocol will be used, where so-called return codes allow voters to verify that their ballots will be counted as cast.

This paper discusses a slightly simplified version of the cryptographic protocol. The description and analysis of the simplified protocol contains most of the ideas and concepts used to build and analyse the full protocol. In particular, the simplified protocol uses the full protocol's novel method for generating the return codes.

The security of the protocol relies on a novel hardness assumption similar to Decision Diffie-Hellman. While DDH is a claim that a random subgroup of a non-cyclic group is indistinguishable from the whole group, our assumption is related to the indistinguishability of certain special subgroups. We discuss this question in some detail.

Keywords: electronic voting protocols, Decision Diffie-Hellman.

1 Introduction

The Norwegian government will run a trial of internet remote voting during the 2011 local government elections. During the advance voting period, voters in 10 municipalities will be allowed to vote from home using their own computers.

Internet voting, and electronic voting in general, faces a long list of security challenges. For Norway, the two most significant security problems with internet voting will be compromised voter computers and coercion.

Coercion will be dealt with by allowing voters to revoke electronically. Revoting cancels previously submitted ballots. Also, the voter may vote once on paper, in which case every submitted electronic ballot is canceled, even those submitted after the paper ballot submission. In theory, almost everyone should therefore have sufficient tools to avoid coercion.

This leaves compromised computers as the main remaining threat. Since a significant fraction of home computers are compromised, the voting system must allow voters to detect ballot tampering without relying on computers. This is complicated by the fact that voters are unable to do even the simplest cryptographic processing without computer assistance.

* Funded in part by the Norwegian Research Council's VERDIKT programme project 183195.

Norwegian municipal elections are somewhat complicated. The voter chooses a party list, he is allowed to give personal votes to candidates on the list, and he is allowed to amend the list by adding a certain number of candidates from other party lists. Essentially then, a ballot consists of a short, variable-length sequence of options (at most about a hundred options) chosen from a small set of possible options (at most a few thousand). Note that the entire sequence is required to properly interpret and count the ballot, but order within the sequence does not matter.

We note in Norway paper ballots submitted in an election are considered sensitive and access is restricted. One reason for this is that because of the complex ballots, many distinct ballots will have essentially the same effect on the outcome. Therefore, it is possible to mark ballots, which means that if the counted ballots were public, anyone could reliably buy votes.

Related work. We can roughly divide the literature into protocols suitable for voting booths [6,7,21,22], and protocols suitable for remote internet voting [1,8,14,16,19], although protocols often share certain building blocks. One difference is that protocols for voting booths should be both coercion-resistant and voter verifiable, while realistic attack models (the attacker may know more than the voter knows) for remote internet voting probably make it impossible to achieve both voter verifiability and coercion-resistance.

For internet voting protocols, we can again roughly divide the literature into two main strands distinguished by the counting method. One is based on homomorphic tallying. Ballots are encrypted using a homomorphic cryptosystem, the product of all the ciphertexts is decrypted (usually using some form of threshold decryption) to reveal the sum of the ballots. For simple elections, this can be quite efficient, but for the Norwegian elections, this quickly becomes unwieldy.

The other strand has its origins in mix nets [4]. Encrypted ballots are sent through a mix net. The mix net ensures that the mix net output cannot be correlated with the mix net input. There are many types of mixes, based on nested encryption [4] or reencryption, verifiable shuffles [11,19] or probabilistic verification [2,14], etc. These can be quite efficient, even for the Norwegian elections.

Much of the literature ignores the fact that a voter simply will not do any computations. Instead, the voter delegates computations to a computer. Unfortunately, a voter's computer can be compromised, and once compromised may modify the ballot before submission.

One approach to defend against compromised computers is so-called preencrypted ballots and return codes [5,3], where the voter well in advance of the election receives a table with candidate names, identification numbers and return codes. The voter inputs a candidate identification number to vote and receives a response. The voter can verify that his vote was correctly received by checking the response against the printed return codes. In Norway, preencrypted ballots will be too complicated, but return codes can still be used.

Note that unless such systems are carefully designed, privacy will be lost. Clearly, general multiparty computation techniques can be used to divide the

processing among several computing nodes (presumably used by [5]). In practice, few independent data centres are available that have sufficient quality and reputation to be used in elections. This means that general multiparty computation techniques are not so useful.

One approach for securely generating the return codes is to use a proxy oblivious transfer scheme [12,13]. A ballot box has a database of return codes and the voter’s computer obviously transfers the correct one to a messenger, who then sends the return code to the voter. The main advantage of this approach is that very few computing nodes are required. Unfortunately, this particular solution is probably too computationally expensive to be used for Norwegian elections.

Another useful tool is the ability for out-of-band communication with voters [17]. This allows us to give the voter information directly, information that his computer should not know and not be able to tamper with. The scheme in [12,13] sends return codes to the voter out-of-band. This helps ensure that a voter is notified whenever a vote is recorded, preventing a compromised computer from undetectably submitting ballots on the voter’s behalf.

Our contribution. The cryptographic protocol to be used in Norway is designed by Scytel, a Spanish electronic voting company, with contributions by the present author. It is mostly a fairly standard internet voting system based on ElGamal encryption of ballots and a mix-net before decryption.

The system works roughly as follows (see Fig. 1). The *voter* V gives his ballot to a *computer* P , which encrypts the ballot and submits it to a *ballot box* B . The ballot box and a *return code generator* R cooperate to compute a sequence of return codes for the submitted ballot. These codes are sent by SMS to the voter’s mobile phone F . The voter verifies the return codes against a list of precomputed option–return code pairs printed on his voting card.

Once the ballot box closes, the submitted ciphertexts are decrypted by a *decryptor* D . An *auditor* A supervises the entire process.

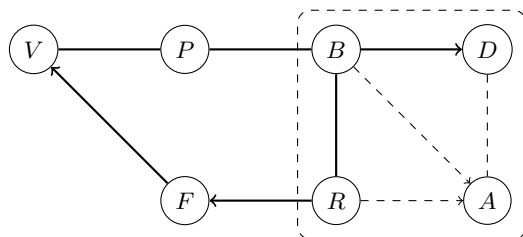


Fig. 1. Overview of the protocol

The main contribution of the current author to the protocol, and the focus of this paper, is a novel method for computing the return codes efficiently. We use the fact that exponentiation is in some sense a pseudo-random function [9,10], and since ElGamal is homomorphic, exponentiation can be efficiently done “inside” the ciphertext.

The mechanics of the Norwegian electoral system means that we must encrypt each option separately in order to generate return codes for each option separately. If every ballot consisted of up to 100 ElGamal ciphertexts, mixing would be prohibitively expensive. Therefore, Scytl uses a clever encoding of options [18,20] to allow the protocol to compress ciphertexts before mixing and then recover the complete ballot after decryption. However, this encoding forces us to do a more careful security analysis of the return code computation.

The main advantage of our contribution is that generating return codes is cheap, amounting to just a few exponentiations. At the same time, mixing is reasonably fast. Since the protocol requires very few players to achieve reasonable security, we have a protocol that is deployable in practice. Indeed, the protocol has already been used for several smaller elections, and will be used in municipal elections August–September 2011.

Overview of the paper. Due to space restrictions, this paper can only present a simplified version of the cryptographic voting protocol used in the 2011 elections. However, the simplified version and its analysis already contains most of the ideas and concepts used in the analysis of the full protocol, and will give the reader a clear idea of how the full protocol works.

The cryptographic protocol is essentially based on ElGamal encryptions. Sect. 2 describes the group structure used for ElGamal and the special properties we require of it to be able to compress many ciphertexts into one, and still recover complete ballots from the decryption. It then defines and discusses a conjecturally hard problem on this group structure, a problem that is similar to Decision Diffie-Hellman and required for the security of the system.

Sect. 3 defines a cryptosystem with corresponding security notions, an instantiation of this cryptosystem, and relates the security of this instantiation to the conjecturally hard problem discussed in Sect. 2. This cryptosystem encapsulates the essential cryptographic operations in the voting protocol. We also briefly discuss the differences between the cryptosystem described here and the corresponding cryptosystem used in the full protocol.

Sect. 4 then shows how we build the cryptographic voting protocol on top of the cryptosystem defined in Sect. 3, and how the security of the voting protocol essentially follows from the security of the cryptosystem. While the voting protocol described is very close to the full voting protocol, it does not achieve as strong security as the full protocol. However, the analysis still shows many of the ideas and techniques used in the analysis of the full protocol.

2 The Underlying Group Structure

Underlying the entire voting protocol is a group structure with certain very specific properties, needed to be able to compress ElGamal ciphertext. The basic idea is that from the product of not too many small primes computed modulo a large prime, the small primes can still be recovered efficiently by trial division. These small primes then lead to a problem similar to Decision Diffie-Hellman.

The Group Structure. Let q be a prime such that $p = 2q + 1$ is also a prime. Then the quadratic residues of the finite field with p elements is a finite cyclic group G of prime order q . Let g be a generator.

Let $\ell_1, \ell_2, \dots, \ell_L$ be the group elements corresponding to the L smallest primes that are quadratic residues modulo p . Let $O = \{1, \ell_1, \ell_2, \dots, \ell_L\}$.

Factoring. Factoring products of small primes is efficient. Suppose that all of these primes are smaller than $\sqrt[k]{p}$ for some integer $0 < k$. Then if we select k elements from O and multiply them, then we can efficiently recover the k elements from the product, up to order. If we use the obvious ordering on the group elements, we get a map from the set of all such products to O^k , and this map can be extended to a map $\phi : G \rightarrow G^k$ by taking any other group element x to the k -tuple $(1, \dots, 1, x)$.

A Related Problem. We are interested in a problem related to the prime p and the elements $\ell_1, \ell_2, \dots, \ell_L$. We begin our discussion with the usual Decision Diffie-Hellman (DDH) problem, which can be formulated as follows:

Decision Diffie-Hellman. Given $(g_1, g_2) \in G \times G$ (where at least g_2 is sampled at random), decide if $(x_1, x_2) \in G \times G$ was sampled uniformly from the set $\{(g_1^s, g_2^s) \mid 0 \leq s < q\}$ or uniformly from $G \times G$.

It is well-known (e.g. [9][10]) that this is equivalent to the following problem:

Alternative DDH. Given $(g_1, \dots, g_L) \in G^L$ (where at least g_2, \dots, g_L are sampled at random), decide if the tuple $(x_1, \dots, x_L) \in G^L$ was sampled uniformly from the set $\{(g_1^s, \dots, g_L^s) \mid 0 \leq s < q\}$ or uniformly from G^L .

However, suppose the subgroup is generated by small primes instead of random group elements. We get the following problem, which we call the Subgroup Generated by Small Primes (SGSP) problem:

Subgroup Generated by Small Primes. Let the prime p be chosen at random from an appropriate range, and let the generator g be chosen at random. Determine the group elements $\ell_1, \ell_2, \dots, \ell_L$ as above. The problem is to decide if $(x_0, x_1, \dots, x_L) \in G^{L+1}$ was sampled uniformly from the set $\{(g^s, \ell_1^s, \dots, \ell_L^s) \mid 0 \leq s < q\}$ or uniformly from G^{L+1} .

While this problem is very similar to Decision Diffie-Hellman, and indeed cannot be hard unless Decision Diffie-Hellman is hard, it seems unlikely that its hardness follows from hardness of Decision Diffie-Hellman.

It is generally believed that the best way to solve the Decision Diffie-Hellman is to compute one of the corresponding discrete logarithms. It is known [15] that solving the static Diffie-Hellman problem with a fairly large number of oracle queries is easier than solving the discrete logarithm problem.

For fairly large L , a static Diffie-Hellman solver could be applied to decide the above problem. This would be faster than the fastest known solver for the Decision Diffie-Hellman problem in the same group. However, for our application, L will always be small, hence the static Diffie-Hellman solver can not be directly

applied. A hybrid approach could perhaps be deployed, but for small L such an approach would not be significantly faster than simply computing a discrete logarithm.

Remark 1. Note that it will probably be possible to choose the prime *together* with a relation among the small primes. Given such a relation, the decision problem above will be easy, since the relation will hold for prime powers as well. It is therefore important for our purposes that the prime p is chosen verifiably at random. There are straight-forward ways to do this.

Further Analysis. While the problem discussed above is sufficient for our purposes, it is not necessary. A weaker, sufficient condition would be if, given a permutation of a subset of $\{\ell_1^s, \dots, \ell_L^s\}$ and g^s for some random s , it was hard to deduce any information about which primes were involved and what the permutation was.

We study the case when there are only two elements, say ℓ_0 and ℓ_1 , and the subset contains one of them. Let A be an algorithm that takes as input five group elements and outputs 0 or 1. Define

$$\begin{aligned}\pi_{00} &= \Pr[A(\ell_0, \ell_1, g, g^s, \ell_0^s) = 0], \\ \pi_{11} &= \Pr[A(\ell_0, \ell_1, g, g^s, \ell_1^s) = 1], \text{ and} \\ \pi_{i,\text{rnd}} &= \Pr[A(\ell_0, \ell_1, g, g^s, g^t) = i], \quad i \in \{0, 1\},\end{aligned}$$

where s and t are sampled uniformly at random from $\{0, 1, \dots, q-1\}$. Note that $\pi_{0,\text{rnd}} = 1 - \pi_{1,\text{rnd}}$, since the input distribution to A is identical for both probabilities.

We may define the advantage of A as $|\pi_{00} + \pi_{11} - 1|$. Observe that if $|\pi_{00} - \pi_{0,\text{rnd}}|$ or $|\pi_{11} - \pi_{1,\text{rnd}}|$ are large, we have a trivial solver for Decision Diffie-Hellman with the generator fixed to either ℓ_0 or ℓ_1 .

We may assume that $\pi_{00} + \pi_{11} - 1 = 2\epsilon > 0$. Then either $\pi_{00} \geq 1/2 + \epsilon$ or $\pi_{11} \geq 1/2 + \epsilon$, so assume the former. Furthermore, let $\pi_{00} - \pi_{0,\text{rnd}} = \mu$. If $|\mu| \geq \epsilon$, we have an adversary against Decision Diffie-Hellman with the generator fixed to ℓ_0 , so assume $|\mu| < \epsilon$. Then

$$\pi_{11} - \pi_{1,\text{rnd}} = 1 + 2\epsilon - \pi_{00} - (1 - \pi_{0,\text{rnd}}) = 2\epsilon - \mu \geq \epsilon,$$

which means that we must have an adversary with advantage at least ϵ against Decision Diffie-Hellman with the generator fixed to either ℓ_0 or ℓ_1 .

The same arguments applies to an algorithm that can recognize one out of multiple elements. It must lead to a successful adversary against Decision Diffie-Hellman with the generator fixed to one of the elements.

Unfortunately, the above argument breaks down if the algorithm is allowed to see multiple elements raised to the same power, that is, if given $\{\ell_i^s \mid i \in I\}$ for some small index set I , the algorithm can decide what I is. It is not unlikely that a careful analysis could reduce this problem to our alternative DDH problem (for a smaller number of primes), but such a result is of questionable value.

3 The Cryptosystem

In order to simplify the analysis of the Norwegian internet voting protocol, we isolate the most essential cryptographic operations into a cryptosystem that can be considered in isolation. We can then later use the security properties of the cryptosystem to reason about the voting protocol's security.

We briefly summarize the relevant cryptographic operations.

Before an election can be run, keys must be generated and the per-voter option–return code correspondence must be set up.

During ballot submission, the voter V 's computer P encrypts the voters' ballot into a ciphertext that is tied to the voter's identity. The ballot box B transforms this ciphertext into a new ciphertext that contains pre-codes, a half-way step between options and return codes. The return code generator R decrypts this ciphertext in order to get the pre-codes, which it will turn into human-readable return codes.

During counting, the ballot box extracts “naked” ciphertexts that cannot be tied to individual voters. The decryptor D decrypts the naked ciphertexts and convinces the auditor A that the decryptions are correct.

3.1 Preliminaries

Let I be a set of identities, M a set of messages and $O \subseteq M$ a set of options, one of which is the null option denoted by 1_O . A ballot is a k -tuple of options. We denote options by v and a ballot (v_1, v_2, \dots, v_k) by \mathbf{v} .

Let C be a set of *pre-codes*, one of which is the null pre-code denoted by 1_C . We shall have a set S of pre-code maps from M to C such that for every $s \in S$, $s(1_O) = 1_C$. We also need a set of commitments to pre-code maps, one commitment for each map.

We extend pre-code maps to k -tuples of messages $\mathbf{m} = (m_1, m_2, \dots, m_k)$ as $s(\mathbf{m}) = (s(m_1), \dots, s(m_k)) \in C^k$.

We also require a total order \prec on the set of options, and a canonical ordering map on ballots $\omega : O^k \rightarrow O^k$ such that for any ballot \mathbf{v} , if $\omega(\mathbf{v}) = (v_1, v_2, \dots, v_k)$, then for any $1 \leq i \leq j \leq k$, $v_i \prec v_j$. This map is extended in some way to a map $\omega : M^k \rightarrow M^k$.

3.2 Definition

Our cryptosystem consists of six algorithms and one protocol:

- A *key generation algorithm* \mathcal{K} that outputs a public key ek , a decryption key dk_1 , a transformation key dk_2 and a code decryption key dk_3 .
- A *pre-code map generation algorithm* \mathcal{S} that on input of a public key ek and an identity V outputs a pre-code map s and a commitment γ to that map.
- An *encryption algorithm* \mathcal{E} that on input of an encryption key ek , an identity $V \in I$ and a message sequence $\mathbf{m} \in M^k$ outputs a ciphertext c .

- A deterministic *extraction algorithm* \mathcal{X} that on input of a ciphertext c outputs a *naked ciphertext* \tilde{c} .
- A *transformation algorithm* \mathcal{T} that on input of a transformation key dk_2 , an identity $V \in I$, a pre-code map s and a ciphertext c outputs a pre-code ciphertext \check{c} or the special symbol \perp .
- A deterministic *pre-code decryption algorithm* \mathcal{D}_R that on input of a pre-code decryption key dk_3 , an identity $V \in I$, a pre-code map commitment γ , a ciphertext c and a pre-code ciphertext \check{c} outputs a sequence of pre-codes $\rho \in C^k$.
- A *decryption protocol* Π_{DP} between a prover and a verifier. The common input is a public key ek and a sequence of naked ciphertexts $\tilde{c}_1, \dots, \tilde{c}_n$. The prover's private input is a decryption key dk_1 . The prover and the verifier output either \perp or a sequence of messages $\mathbf{m}_1, \dots, \mathbf{m}_n$.

Such a cryptosystem cannot be useful unless it guarantees correct decryption of ciphertexts and transformed ciphertexts. We capture this with the following completeness requirements:

- C1. *For any sequence of messages, encrypting, extracting and then running the decryption protocol should faithfully reproduce the messages, up to the action of the order map.*

For any message and identity sequences $\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_n$ and V_1, V_2, \dots, V_n , if the following actions happen:

$$(ek, dk_1, dk_2, dk_3) \leftarrow \mathcal{K}; \text{ for } i \text{ from } 1 \text{ to } n: c_i \leftarrow \mathcal{E}(ek, V_i, \mathbf{m}_i), \tilde{c}_i \leftarrow \mathcal{X}(c_i); \text{ the protocol } \Pi_{\text{DP}} \text{ is run with } ek \text{ and } (\tilde{c}_1, \dots, \tilde{c}_n) \text{ as public input and } dk_1 \text{ as the prover's private input.}$$

Then the prover and verifier in the protocol both output the same sequence of messages, and that sequence is a permutation of the sequence $\omega(\mathbf{m}_1), \dots, \omega(\mathbf{m}_n)$.

- C2. *Transformation of a ciphertext should apply the given pre-code map to the content of the ciphertext.*

For any $\mathbf{m} \in M^k$ and any $V \in I$, if the following actions happen:

$$(ek, dk_1, dk_2, dk_3) \leftarrow \mathcal{K}; (s, \gamma) \leftarrow \mathcal{S}(ek, V); c \leftarrow \mathcal{E}(ek, V, \mathbf{m}); \check{c} \leftarrow \mathcal{T}(dk_2, V, s, c); \rho \leftarrow \mathcal{D}_R(dk_3, V, \gamma, c, \check{c}).$$

Then $\check{c} \neq \perp$ and $\rho = s(\mathbf{m})$.

3.3 Security Requirements

We define a set of fairly natural notions of security for the cryptosystem, relating to privacy and integrity.

D-Privacy. *Naked ciphertexts should not be correlatable to identities.*

For any $V \in I$ and $\mathbf{m} \in M^k$, if the following actions happen:

$$(ek, dk_1, dk_2, dk_3) \leftarrow \mathcal{K}; c \leftarrow \mathcal{E}(ek, V, \mathbf{m}); \tilde{c} \leftarrow \mathcal{X}(c).$$

Then the distribution of \tilde{c} should be independent of V .

B-Privacy. *An adversary that knows the transformation key should not be able to say anything about the content of any ciphertexts he sees.* We play the following game between a simulator and an adversary, and the probability that the adversary wins should be close to $1/2$.

A simulator samples $b \leftarrow \{0, 1\}$ and computes $(ek, dk_1, dk_2, dk_3) \leftarrow \mathcal{K}$. The simulator also chooses a sequence of random messages $\mathbf{m}_1^1, \mathbf{m}_2^1, \dots, \mathbf{m}_n^1$. The adversary gets ek and dk_2 , and sends a sequence of challenge messages $\mathbf{m}_1^0, \mathbf{m}_2^0, \dots, \mathbf{m}_n^0$ to the simulator, one by one, along with identities V_1, V_2, \dots, V_n .

When the simulator gets (\mathbf{m}_i^0, V_i) , $1 \leq i \leq n$, it computes $c_i \leftarrow \mathcal{E}(ek, V_i, \mathbf{m}_i^b)$ and sends c_i to the adversary.

Finally, the adversary outputs $b' \in \{0, 1\}$ and wins if $b = b'$.

R-Privacy. *An adversary that controls the pre-code decryption key and sees many transformed encryptions of valid ballots from O^k should not be able to say anything non-trivial about the content of those encryptions.* We play the following game between a simulator and an adversary, and the probability that the adversary wins should be close to $1/2$.

A simulator samples $b \leftarrow \{0, 1\}$, a random permutation π_1 on O and sets π_0 to be the identity map on O . It computes $(ek, dk_1, dk_2, dk_3) \leftarrow \mathcal{K}$. The adversary gets ek and dk_3 , and chooses a challenge identity V . The simulator computes $(s, \gamma) \leftarrow \mathcal{S}(ek, V)$ and sends γ to the adversary.

The adversary then submits a sequence of ballots $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n$ from O^k , one by one. The simulator computes $c_i \leftarrow \mathcal{E}(ek, V, \pi_b(\mathbf{v}_i))$, $\tilde{c} \leftarrow (\mathcal{T}(dk_2, V, s, c_i))$ and sends (c_i, \tilde{c}_i) to the adversary.

Finally, the adversary outputs $b' \in \{0, 1\}$ and wins if $b = b'$.

A-Privacy. *An honest-but-curious adversary that runs the verifier part of the decryption protocol should not be able to correlate ciphertexts with decryptions.* We play the following game between a simulator and an adversary, and the probability that the adversary wins should be close to $1/2$.

A simulator samples $b \leftarrow \{0, 1\}$ and computes $(ek, dk_1, dk_2, dk_3) \leftarrow \mathcal{K}$. The adversary gets ek , then chooses a sequence of identities V_1, \dots, V_n and messages $\mathbf{m}_1, \dots, \mathbf{m}_n$.

The simulator sets π_0 to be the identity map on $\{1, 2, \dots, n\}$, and samples a random permutation π_1 on $\{1, 2, \dots, n\}$. Then the simulator computes $c_i \leftarrow \mathcal{E}(ek, V_i, \mathbf{m}_{\pi_b(i)})$, $\tilde{c}_i \leftarrow \mathcal{X}(c_i)$ for $i = 1, 2, \dots, n$, sends c_1, \dots, c_n to the adversary and runs the prover part of the protocol Π_{DP} with appropriate input against the adversary's verifier.

Finally, the adversary outputs $b' \in \{0, 1\}$ and wins if $b = b'$.

P-Integrity. *An adversary that knows the public key ek should not be able to create an identity, a ballot and a ciphertext such that the transformed ciphertext decryption is consistent with the ballot, but the decryption of the ciphertext is inconsistent with the ballot.* We play the following game between a simulator and an adversary, and the probability that the adversary wins should be close to 0.

A simulator computes $(ek, dk_1, dk_2, dk_3) \leftarrow \mathcal{K}$. The adversary gets ek , then produces a tuple (V, \mathbf{v}, c) . The simulator computes $(s, \gamma) \leftarrow \mathcal{S}(ek, V)$, $\rho \leftarrow$

$\mathcal{D}_R(dk_3, V, \gamma, c, \tilde{c})$, $\tilde{c} \leftarrow \mathcal{X}(c)$ and runs both parts of the protocol Π_{DP} on the public input \tilde{c} and appropriate private input to get the decryption \mathbf{m} .

The adversary wins if the simulator's computation completes without error and $s(\mathbf{v}) = \boldsymbol{\rho}$, while $\omega(\mathbf{v}) \neq \omega(\mathbf{m})$.

Remark 2. The cryptosystem is a convenient abstraction of one part of the voting protocol. Sect. 4 shows that security of the voting protocol follows from the above security properties and other security measures in the protocol.

Remark 3. Recall that we only have space for proving a weak notion of security for the voting system. The security notions *B-Privacy*, *A-Privacy* and *P-Integrity* have therefore been weakened to simplify the presentation. The full security analysis also requires a form of plaintext awareness for the encryption and a notion of *D-Integrity*, which are not discussed here.

3.4 Instantiation

We shall now describe an instantiation of the above cryptosystem, which is a simplified version of the instantiation that will be deployed in the Norwegian trials. It will be based on the group structure described in Sect. 2.

The set of group elements of G will be both the message space M and the set of pre-codes C . We interpret O as the set of options, and 1 as the null option and null code.

The set of pre-code maps S is the set of automorphisms on G , which corresponds to the set of exponentiation maps $\{x \mapsto x^s \mid s \in \{1, 2, \dots, q-1\}\}$. We commit to a pre-code map s by computing $s(g) \in G$.

We define the map ω as

$$\omega(\mathbf{m}) = \phi(m_1 m_2 \cdots m_k).$$

- The *key generation algorithm* \mathcal{K} samples a_1 and a_2 uniformly at random from $\{0, 1, \dots, q-1\}$, then computes $a_3 = a_1 + a_2 \bmod q$, $y_1 = g^{a_1}$, $y_2 = g^{a_2}$ and $y_3 = g^{a_3}$. The public key is $ek = (y_1, y_2, y_3)$. The decryption key is $dk_1 = a_1$, the transformation key is $dk_2 = a_2$ and the code decryption key is $dk_3 = a_3$.
- The *pre-code map generation algorithm* $\mathcal{S}(ek, V)$ samples s uniformly from the set $\{1, 2, \dots, q-1\}$. It computes $\gamma = g^s$ and outputs the map determined by s and the commitment γ .
- The *encryption algorithm* $\mathcal{E}(ek, V, \mathbf{v})$ samples t_1, t_2, \dots, t_k uniformly at random from $\{0, 1, 2, \dots, q-1\}$, then computes $(x_i, w_i) = (g^{t_i}, y_1^{t_i} v_i)$ for $i = 1, 2, \dots, k$. The ciphertext is $c = ((x_1, w_1), (x_2, w_2), \dots, (x_k, w_k))$.
- The *extraction algorithm* $\mathcal{X}(c)$, $c = ((x_1, w_1), \dots, (x_k, w_k))$, computes $\tilde{x} = x_1 x_2 \cdots x_k$ and $\tilde{w} = w_1 w_2 \cdots w_k$, then outputs the naked ciphertext $\tilde{c} = (\tilde{x}, \tilde{w})$.
- The *transformation algorithm* $\mathcal{T}(dk_2, V, s, c)$, $c = ((x_1, w_1), \dots, (x_k, w_k))$, computes $(\tilde{x}_i, \tilde{w}_i) = (x_i^s, (w_i x_i^{a_2})^s)$ for $i = 1, 2, \dots, k$, and outputs the pre-code ciphertext $\check{c} = ((\tilde{x}_1, \tilde{w}_1), \dots, (\tilde{x}_k, \tilde{w}_k))$.

- The *pre-code decryption algorithm* $\mathcal{D}_R(dk_3, V, \gamma, c, \check{c})$, with $\check{c} = ((\tilde{x}_1, \tilde{w}_1), \dots, (\tilde{x}_k, \tilde{w}_k))$, computes $\rho_i = \tilde{w}_i \tilde{x}_i^{-dk_3}$ for $i = 1, 2, \dots, k$ and outputs the precode $\boldsymbol{\rho} = (\rho_1, \dots, \rho_k)$.
- The simplified *decryption protocol* is a trivial protocol. The prover gets as input a_1 and $\tilde{c}_1, \tilde{c}_2, \dots, \tilde{c}_n$, where $\tilde{c}_i = (\tilde{x}_i, \tilde{w}_i)$. It selects a random permutation π on the set $\{1, 2, \dots, n\}$ and computes $\tilde{m}_{\pi(i)} = \tilde{w}_i \tilde{x}_i^{a_1}$, which it shares with the verifier. They then separately compute $\mathbf{m}_j = \phi(\tilde{m}_j)$ for $i = 1, 2, \dots, n$ and output the result.

The first completeness requirement **C1** is satisfied, because ElGamal encryptions are homomorphic and the map ϕ recovers a proper representation of the ballot from the product. The second completeness requirement **C2** is again satisfied because ElGamal is homomorphic. For an encryption of m and a pre-code map s , we get that $\tilde{x} = g^{ts}$ and

$$\tilde{w} = w^s x^{a_2} = g^{tsa_1} m^s g^{tsa_2} = g^{ts(a_1+a_2)} m^s = g^{tsa_3} m^s = y_3^{ts} m^s.$$

We argue briefly for security, under the assumption that the problem described in Sect. **2** is hard, which also implies that Decision Diffie-Hellman is hard.

D-Privacy. Since the voter identity is not used to create the ciphertext, this holds trivially. (This will essentially be true for the full protocol too.)

B-Privacy. The encryption algorithm is essentially parallel ElGamal encryption. Since ElGamal is secure if Decision Diffie-Hellman is hard, we have *B-Privacy*.

R-Privacy. For any one voter, the adversary cannot decide if the pre-code ciphertexts contain v^s for the various options $v \in O$, or values t^s chosen at random, one for each option. It follows that the adversary cannot decide if the options are permuted or not.

This holds as long as the SGSP problem described in Sect. **2** is hard. The proof is fairly straight-forward, but somewhat technical, so we omit it.

A-Privacy. The naked ciphertexts that the honest-but-curious adversary gets are just ElGamal encryptions of the ballots. Since Decision Diffie-Hellman is hard, ElGamal will be secure and the adversary cannot distinguish encryptions of the real ballots from encryptions of random nonsense. It then follows that the adversary cannot decide if the ballots are permuted before encryption or not.

P-Integrity. Since the transformation algorithm essentially applies an automorphism to the content of the ciphertext, *P-Integrity* follows trivially.

Remark 4. The full cryptosystem is an extension of this system. The encryption algorithm adds a proof of knowledge of decryption to ElGamal ciphertexts that also ties the voter's identity to the ciphertext. The transformation algorithm verifies this proof, and itself includes a proof of correct computation in the pre-code ciphertext. The pre-code decryption algorithm verifies both proofs before decryption. The decryption protocol first runs a verifiable mix-net, then verifiably decrypts the shuffled ciphertexts.

The proof of knowledge essentially provides a form of plaintext awareness, and also ties the ciphertext to the voter's identity. The proof of correct computation reduces the opportunities a corrupt ballot box has to deviate from honest-but-curious behaviour. The mix-net and verifiable decryption hide the relationship between the ciphertexts and their decryptions from the verifier without allowing the prover to cheat.

4 The Voting Protocol

The voting protocol is built on top of the cryptosystem from Sect. 3, together with several other tools.

The players in the voting protocol are the voters, the voters' computers, the voters' telephones, the electoral board and four infrastructure players: a ballot box B , a return code generator R , a decryptor D and an auditor A .

4.1 Assumptions about the Environment

We idealize the environment in which the voting protocol is deployed. We assume that each voter V has his own personal computer P_V and personal phone F_V , that there are secure, identified and authenticated channels between the voters' computers and the ballot box, that there are secure channels between the infrastructure players, and that there is a secure one-way channel from the return code generator to the voters' phones.

We shall assume the existence of an ideal PKI such that each computer can digitally sign on behalf of its voter and that every infrastructure player can verify such signatures. We shall also assume that the return code generator has a signing key and that every computer has the corresponding verification key.

The only functionality we require of the telephones is the ability to receive messages from the return code generator and show these messages to the voters. We shall also assume that delivery of these messages are under the adversary's control, subject to the requirement that before the voter begins a new ballot submission process, any pending messages to the phone should be processed and delivered to the voter.

All of these assumptions and idealizations are reasonable, but lack of space prevents further discussion.

4.2 Additional Cryptography

The return code generator will need a random-looking function from $I \times C$ into a (small) set of human-readable codes C_H . Therefore, we shall assume that we have a pseudo-random function family \mathcal{F} of function from $I \times C$ to C_H , subject to the requirement that the blank pre-code is taken to a special blank human-readable code.

Furthermore, a collision resistant hash function is needed.

4.3 Key Generation

Since our focus in this paper is not on the key generation, we shall assume that all key generation is done by a trusted dealer. This trusted dealer does as follows:

$$(ek, dk_1, dk_2, dk_3) \leftarrow \mathcal{K}; d \leftarrow \mathcal{F}; \text{ for each voter } V, (s_V, \gamma_V) \leftarrow \mathcal{S}(ek, V).$$

The trusted dealer sends the generated keys to the players as detailed in Table 1.

Table 1. Distribution of key material to the players

Player	Key material
V	The set $\{(m, d(V, s_V(m))) \mid m \in \mathcal{O} \setminus \{1_O\}\}$.
P	The public key ek .
B	The transformation key dk_2 and the set $\{(V, s_V)\}$.
R	The pre-code decryption key dk_3 , the pseudo-random function d and the set $\{(V, \gamma_V)\}$.
D	The decryption key dk_1 .
A	The public key ek .

4.4 Protocol

After key generation, the protocol runs in two phases: ballot submission and counting. During the submission phase, only the voters, their computers and phones, the ballot box and the return code generator are active. During the counting phase, only the infrastructure players are active.

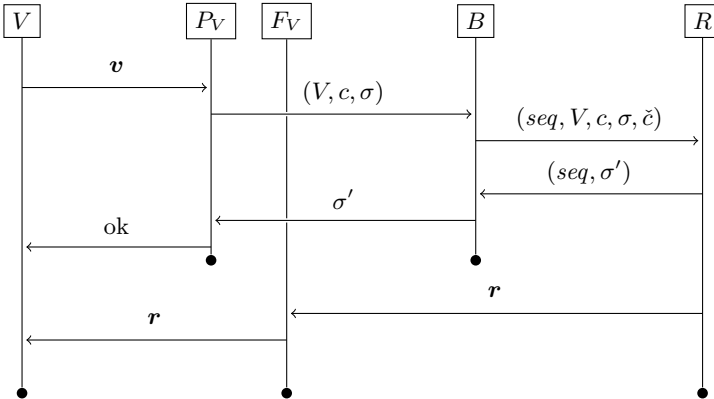


Fig. 2. Overview of messages sent during ballot submission

Submission Phase During the submission phase, the players do as follows (as summarized in Fig. 2):

- The *voter* V will not submit multiple ballots in parallel. When he wants to submit a ballot \mathbf{v} , he sends the ballot to his computer. He then waits for the computer to accept the ballot as cast and the phone to display the message \mathbf{r} . The voter accepts the ballot as cast if for $i = 1, 2, \dots, k$, if $v_i = 1_O$, then r is the blank code, otherwise (v_i, r_i) is in the set he received from the trusted dealer. If any of the above steps fail, the voter does not accept the ballot as cast.

If the voter's phone ever displays a message \mathbf{r} when the voter is not submitting a ballot, the voter will complain about a forgery.

- On input \mathbf{v} from V , the *computer* P_V computes $c \leftarrow \mathcal{E}(ek, V, \mathbf{v})$, computes a signature σ on c and sends (V, c, σ) to the ballot box. It waits for σ' from the ballot box and verifies that σ' is the return code generator's signature on a hash of V and c .
- On input \mathbf{r} from R , the voter's *phone* sends \mathbf{r} to the voter.
- On input (V, c, σ) from P_V , the *ballot box* temporarily blocks ballot submissions by V and chooses the next sequence number seq . It then computes $\check{c} \leftarrow \mathcal{T}(dk_2, V, s_V, c)$ and sends $(seq, V, c, \check{c}, \sigma)$ to the return code generator. Then it waits for the reply (seq, σ') from the return code generator, in which case it verifies that σ' is the return code generator's signature on a hash of V and c , records (seq, V, c) and sends σ' to P_V .

Finally, it removes the block on ballot submission by V .

- On input $(seq, V, c, \check{c}, \sigma)$ from the ballot box, the *return code generator* blocks any further processing of submissions from V , checks that it has not seen c before, nor a ballot submission for V with a sequence number higher than or equal to seq . Then it computes $\rho \leftarrow \mathcal{D}_R(dk_1, V, \gamma, c, \check{c})$, and a signature σ' on a hash of V and c . For $i = 1, 2, \dots, k$, if $\rho_i = 1_C$, then r_i is the blank human-readable code, otherwise $r_i = d(V, \rho_i)$.

The return code generator then records that it has seen c and (seq, V, c, σ) , sends (seq, σ') to B , sends \mathbf{r} to the voter's phone F_V , and removes the block on processing submissions by V .

Counting Phase. During the counting phase, the players do as follows:

- The *ballot box* selects for each voter the ciphertext c with the highest sequence number and extracts the naked ciphertext by computing $\tilde{c} \leftarrow \mathcal{X}(c)$. It then sorts the resulting ciphertexts, resulting in a sequence of naked ciphertexts $\tilde{c}_1, \dots, \tilde{c}_n$, which is sent to D .

Finally, the ballot box sends every recorded submission to the auditor A .

- The *return code generator* sends a hash of everything it has seen, along with the corresponding identities and sequence numbers to the auditor A .
- On input $\tilde{c}_1, \dots, \tilde{c}_n$ from the ballot box, the *decryptor* runs the prover part of the protocol Π_{DP} with appropriate public and private input against the auditor's verifier run. If the protocol run produces a sequence of plaintext ballots, the decryptor outputs this sequence.

- The *auditor* receives the contents of the ballot box and the return code generator’s records. It verifies that the content of the ballot box matches the return code generator’s records, specifically that they agree on sequence numbers. It extracts naked ciphertexts exactly as the ballot box does, resulting in a sequence of naked ciphertexts $\tilde{c}_1, \dots, \tilde{c}_n$, then runs the verifier part of the protocol with appropriate public input against the decryptor’s prover run. If the protocol run produces a sequence of plaintext ballots, the auditor outputs this sequence.

4.5 Security Analysis

The simplified protocol does not achieve as strong security as the full protocol. However, to illustrate that even such a weak protocol can achieve interesting security properties, we prove a number of results for a somewhat weak adversary.

We shall consider a very restricted adversary model, described by the following two disjoint cases:

- A1. A subset of the voters’ computers are corrupt.
- A2. Exactly one infrastructure player is honest-but-curious (he follows the protocol, but shares his knowledge with the adversary), every other player is honest.

We make the following (weak) security claims, all under the assumption that keys are all honestly generated:

- S1. At most one ballot per voter will be counted. The number of ballots counted will not be higher than the number of voters who submitted a ballot, attempted to submit a ballot or complained about a forgery.
- S2. If the voter accepts his ballot as cast, it is counted as cast except with small probability, unless the voter later revotes or complains about a forgery.
- S3. If the voter’s computer and the return code generator are both honest, the content of the voter’s ballot remains private.
- S4. If the return code generator is honest-but-curious, the adversary learns the number of null options in each ballot submission, and if a voter submits multiple ballots, can learn where these ballots differ.

Note that when the voter’s computer is corrupt, privacy is lost.

The claim **S1** holds trivially for both adversary models.

The claim **S2** holds trivially for **A2**. We need to consider what happens when the voter’s computer is corrupt.

The voter does not accept the ballot as cast unless he receives the correct human-readable return code. If the return code generator receives the wrong pre-code, the pseudo-random function ensures that the voter gets the wrong human-readable return code, except with small probability.

By P -Integrity of the cryptosystem, the return code generator will get the wrong pre-code unless the corrupt computer has encrypted the intended ballot.

In other words, a voter will incorrectly accept a ballot as cast only with small probability. Once a ballot has been accepted as cast, the formal requirements on

the cryptosystem ensures that it will be counted as cast, unless further ballots are submitted.

Furthermore, if the computer submits a ballot on the voter's behalf without the voter's knowledge, the voter will receive a human-readable return code and complain about a forgery.

Consider the claim **S3**. If the *ballot box* is honest-but-curious, this follows directly from B -Privacy of the cryptosystem. The honest-but-curious ballot box will receive the encryptions of the ballots, but cannot distinguish them from encryptions of random nonsense. It then follows that he cannot deduce anything about the content of the ciphertexts.

The honest-but-curious *decryptor* sees the naked ciphertexts and knows the decryption key. The distribution of each naked ciphertext is statistically independent from the identity that was used to create it. That is, the naked ciphertext depends only on the message it contains and the randomness used, nothing else. It then follows that the naked ciphertexts are independent of the order in which they were created.

This means that once the ciphertexts are sorted by the ballot box, the resulting order is independent of the order in which the ciphertexts were created. Which in turn means that there is no correlation between voter identities and naked ciphertexts, beyond the fact that the ciphertext contains the voter's ballot.

The honest-but-curious *auditor* learns the contents of the ballot box and inspects the decryption of the naked ciphertexts. The claim then follows directly from A -Privacy. An adversary that runs the verifier part of the protocol and controls the identities and the messages to be encrypted, cannot decide if the ciphertext order has been permuted or not. This implies that when the adversary does not control all of the messages to be encrypted, he still cannot correlate decrypted ballots and voters.

The claim **S4** follows from R -Privacy of the cryptosystem. The honest-but-curious return code generator cannot decide if he sees encryptions of chosen options or random permutations of these chosen options. It follows that the attacker cannot deduce anything about the content of the ciphertexts, beyond identifying where two submitted ballots differ (in which case, the pre-codes should differ).

Note that the return code generator cannot decide if two voters submit the same ballot, since the voters have distinct, personal pre-code maps.

5 Conclusion

We have discussed a new cryptographic problem related to the Decision Diffie-Hellman problem. We have described a simplified version of the cryptographic voting protocol that will be used in the Norwegian government's e-voting experiment in 2011 and analysed its security based on our new cryptographic problem.

While this protocol is quite efficient and practical, we stress that it is possible to improve performance even further by using multi-ElGamal instead of ElGamal. While there are some technical problems, such a change yields a significant performance improvement without reducing security.

Acknowledgements. The author would like to thank Kjell Jørgen Hole, the E-VALG 2011 people, the people at Scytl, Helger Lipmaa, Filip van Laenen, David Wagner, Mariana Raykova and Berry Schoenmakers, as well as many others, for useful discussions and feedback.

References

1. Cohen [Benaloh], J.D., Fischer, M.J.: A robust and verifiable cryptographically secure election scheme (extended abstract). In: Proceedings of 26th Symposium on Foundations of Computer Science, pp. 372–382. IEEE (1985)
2. Boneh, D., Golle, P.: Almost entirely correct mixing with applications to voting. In: Atluri, V. (ed.) ACM Conference on Computer and Communications Security, pp. 68–77. ACM (2002)
3. e-voting security study. CESG, United Kingdom, Issue 1.2. (July 2002)
4. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM* 24(2), 84–88 (1981)
5. Chaum, D.: Surevote (2000), <http://www.surevote.com>
6. Chaum, D.: Secret-ballot receipts: True voter-verifiable elections. *IEEE Security & Privacy* 2(1), 38–47 (2004)
7. Chaum, D., Ryan, P.Y.A., Schneider, S.: A Practical Voter-Verifiable Election Scheme. In: De Capitani di Vimercati, S., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 118–139. Springer, Heidelberg (2005)
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. Damgård, I., Dupont, K., Pedersen, M.Ø.: Unclonable Group Identification. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 555–572. Springer, Heidelberg (2006)
10. Gjøsteen, K.: A Latency-Free Election Scheme. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 425–436. Springer, Heidelberg (2008)
11. 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)
12. Heiberg, S., Lipmaa, H., van Laenen, F.: On achieving e-vote integrity in the presence of malicious trojans. Submission to the Norwegian e-Vote 2011 tender (August 2009), <http://eprint.iacr.org/2010/195>
13. Heiberg, S., Lipmaa, H., van Laenen, F.: On E-Vote Integrity in the Case of Malicious Voter Computers. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) ESORICS 2010. LNCS, vol. 6345, pp. 373–388. Springer, Heidelberg (2010)
14. Jakobsson, M., Juels, A., Rivest, R.L.: Making mix nets robust for electronic voting by randomized partial checking. In: Boneh, D. (ed.) USENIX Security Symposium, pp. 339–353. USENIX (2002)
15. Joux, A., Lercier, R., Naccache, D., Thomé, E.: Oracle-Assisted Static Diffie-Hellman Is Easier Than Discrete Logarithms. In: Parker, M.G. (ed.) Cryptography and Coding 2009. LNCS, vol. 5921, pp. 351–367. Springer, Heidelberg (2009)
16. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. *Cryptology ePrint Archive*, Report 2002/165 (2002), <http://eprint.iacr.org/>
17. Kutyłowski, 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)

18. Naccache, D., Stern, J.: A New Public-Key Cryptosystem. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 27–36. 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. Peng, K.: A Hybrid E-Voting Scheme. In: Bao, F., Li, H., Wang, G. (eds.) ISPEC 2009. LNCS, vol. 5451, pp. 195–206. Springer, Heidelberg (2009)
21. Ryan, P.Y.A., Peacock, T.: Prêt à voter: a systems perspective. Technical Report CS-TR No 929, School of Computing Science, Newcastle University (September 2005)
22. 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)

Transparency and Technical Measures to Establish Trust in Norwegian Internet Voting

Oliver Spycher^{1,2}, Melanie Volkamer³, and Reto Koenig^{1,2}

¹ Bern University of Applied Sciences, CH-2501 Biel, Switzerland
{oliver.spycher,reto.koenig}@bfh.ch

² University of Fribourg, CH-1700 Fribourg, Switzerland
{oliver.spycher,reto.koenig}@unifr.ch

³ TU Darmstadt, CASED, Mornewegstrasse 34, D-64293 Darmstadt
melanie.volkamer@cased.de

Abstract. The short history of e-voting has shown that projects are doomed to fail in the absence of trust among the electorate. The first binding Norwegian Internet elections are scheduled for fall 2011. Notably, transparency is taken as a guideline in the project. This article discusses transparency and other measures the Norwegians apply that are suited to establish profound trust, i.e. trust that grounds on the system's technical features, rather than mere assertions. We show whether at all, how and to which degree these measures are implemented and point out room for enhancements. We also address general challenges of projects which try to reach a high level of transparency for others as lessons learned.

1 Introduction

Voting technology comes with many promises. While some see the potential of saving significant time and money, others will hope to increase voter turnout due to easy and flexible participation. Also, voters may expect mechanisms to validate their ballot and avoid casting an invalid vote. Although many stakeholders in voting are likely to benefit from such features in some way, voting technology still faces much opposition. Correspondingly, only few countries introduced internet voting and so far only Estonia has offered their citizens to vote through the internet at nation-wide parliamentary elections. Critics most commonly express their doubts regarding the integrity of the outcome of elections, arguing that citizens need to be able to verify the correct functioning of the electronic procedure based on the processed data. Accordingly, mistrust towards the Irish voting machines culminated in the cancelation of the respective project shortly before going live. For the same reason Germany and the Netherlands have persistently banned their voting machines from use at political elections. Trust in voting technology that lasts can only be established when operating a system that complies with high security standards. On the other hand, even the perfectly secure system alone will hardly increase any trust among the public. In order to avoid the fate of the voting machines in Germany, the Netherlands and Ireland, we must not only ask ourselves how to make systems that are more

secure. The focus should rather lie on the primary, superordinate question of how to establish trust itself.

Soon we may add Norway to the list of countries offering internet voting for governmental elections. The Norwegian government’s motivation is to increase the availability of the voting system and to reduce costs in the long term. Binding trials of internet voting are planned in ten municipalities for the 2011 local elections in September. Afterwards, the parliament will decide whether to continue the project and enable remote electronic voting for federal elections in the future. The e-voting project called E-valg is currently widely discussed and particularly special due its open information policy. Their motivation for transparency is to increase trust among experts but also among voters in general.

We have taken advantage of the open information policy and expose some implemented techniques that are suited for establishing trust as per [27]. [27] identifies appropriate measures to establish trust not only among experts but also among a non-technical audience. These measures comprise: *transparency*, *implementing separation of duty*, *enabling verifiability*, *enabling vote updating*, *evaluating the system according to international standards* and *test elections*. This set strongly grounds on the ‘Guidelines on transparency of e-enabled elections’ published by the Council of Europe in [19], the discussion in [26] on advantages and disadvantages evaluation and certification versus verifiability, and the recommendation on the information that an e-voting project should publish in [24]. The objective of this paper is twofold: On one hand we validate the recommended measures meant to establish trust [27], by analysing a concrete Internet voting project. At the same time we reveal whether and to which degree the Norwegian Internet voting project adheres to these recommendations.

After providing the necessary background on Norwegian Internet voting in Section 2, we will revisit and briefly introduce each measure in the subsequent sections. We summarize how these measures are addressed and expose room for enhancements where it seems applicable.

2 The Norwegian Internet Voting Project and System

The main manufacturer of the voting system is Scytl, a company specialized in Internet voting solutions. The Norwegian project distinguishes itself from other electronic voting projects in many ways: for instance they put an emphasis on verifiability, they address the untrustworthiness of home computers, publish system specifications, security analyses, the source code and many other project related documents. Furthermore, they are working towards an evaluation according to the Common Criteria [11].

2.1 System Description from the Voters’ Perspective

The voters’ perspective of the system is rather simple and convenient. They authenticate using their MinID¹ account (two-level authentication). After placing

¹ MinID is a well established service in Norway. It can be used to access more than 50 online services from various Norwegian public agencies [11].

the right mouse clicks to select their preferred parties and candidates and cast their vote, they receive an SMS containing personalized verification codes corresponding with the vote received by the voting servers (one code representing the selected party and one code per candidate position of the party list). If the codes correlate with the expected codes of the voting card they previously received by postal mail, voters may be confident that their vote has reached the servers as intended.

2.2 System Entities

The system comprises the following key components.

1. Voter's Computer: Downloads and runs the voting client application (Java Applet).
2. Electoral Roll (ERoll) Service: Holds information on the electorate.
3. Authentication Service: Holds and sends voter credentials to the voter's computer (upon successful authentication by MinID). The credentials include a private key for signing votes.
4. Vote Collector Service (VCS): Stores encrypted votes.
5. Return Code Generator (RCG): Computes the information required by the voter to verify the correct inclusion of her vote in the ballot box. She receives that information by SMS.
6. Key Management Service (KMS): Creates and distributes keys. It is also in charge of establishing the private key used for decrypting the votes. The keys for VCS are generated on KMS-VCS, the ones for RCG on KMS-RCG.
7. Cleansing Service: Discards electronic votes (e-votes) of voters that cast a paper vote (p-vote).
8. Mix-Net (as described in [18]): Mixes and re-encrypts the encrypted votes signs the output. The mix-net consists of four nodes that form a LAN.
9. Decryption / Counting Service: Decrypts and counts the votes.

Except for the ERoll service (developed by Ergo), they all run software developed by Scytl.

The main task of the Electoral Board (EB) is to securely store shared of the decryption key and to initiate the Decryption / Counting Service. There are also auditors involved to verify the integrity of the ballot. The RCG is operated by The Directorate for Civil Protection and Emergency Planning (DSB), which is subordinate to The Ministry of Justice and located in Tønsberg (about 100 km from Oslo and 700 km from Brønnøysund), the VCS (and the rest of the online system) is operated by the Brønnøysund Registry Centre, which is subordinate to The Ministry of Trade and Industry. The isolated components of the system (KMS, cleansing service, mix-net and decryption service) are located in the Crisis Support Unit, a high security facility subordinate to The Ministry of Justice and located in Oslo, but the servers are managed by security cleared representatives of The Ministry of Local Government and Regional Development (KRD).

2.3 Setup

The following steps are conducted prior to the voting phase.

1. Establish Electoral Register: Eligible voters are included in the ERoll.
2. Key Generation:
 - (a) The KMS generates and distributes RSA key pairs for all components to sign messages.² Voters' key-pairs denoted as (h_{voter}, s_{voter}) are saved in the KMS.
 - (b) It also generates ElGamal key pairs for VCS (h_{VCS}, s_{VCS}) , for RCG (h_{RCG}, s_{RCG}) and for EB (h_{EB}, s_{EB}) . These are used with respect to encrypting, decrypting and partially encrypting and decrypting votes (voters encrypt their vote using the public key of the electoral board h_{EB}). The private keys must satisfy $s_{VCS} + s_{EB} = s_{RCG}$, accordingly the public keys $h_{VCS} \cdot h_{EB} = h_{RCG}$. VCS and RCG receive their keys from the KMS through a secure channel (each of the two private keys is generated using an individual isolated machine denoted as KMS-VCS and KMS-RCG respectively). Each electoral board member presents one personal smartcard to KMS-VCS to obtain s_{VCS_i} and the other personal smartcard to KMS-RCG to obtain s_{RCG_i} . The values s_{VCS_i} and s_{RCG_i} are chosen such that a majority of all shares suffice to compute s_{EB} (t, n - threshold). The KMS does not persistently save any private keys. The containing storage media are destroyed after the setup phase.
 - (c) Finally, the KMS generates private symmetric keys K_{VCS} and K_{RCG} and sends them to VCS and RCG through a secure channel. VCS and RCG decrypt and save these values in the private partition of secure hardware (HSM). Again, the values are separately generated in KMS-VCS and KMS-RCG and the storage media destroyed after the setup phase.
3. Establish public parameters: Election specific public values are established in KMS, including values P (the global value associated with a party), C (the global value associated with a candidate), and T (the global value associated with a candidate position in his party list). These values are chosen within the ElGamal plaintext space.
4. Establish Return Codes (KMS-VCS):
 - (a) For each voter, using K_{VCS} and the voter's social security number (SSID), KMS-VCS computes a secret value s .
 - (b) For each voter and each of the values P and C , using s KMS-VCS computes the partial values of the long party codes P' as P^s and the partial values of the long candidate codes C' as C^s .³ P is the global value associated with a party, C the one associated with a candidate. The values P' and C' are sent to KMS-RCG along with information which voter they correspond to.

² Outgoing messages are always signed and the signature of incoming messages always verified throughout the voting procedure, even if not explicitly stated here.

³ Note, that given P' (C'), P (C) can only be computed when knowing s and s cannot be computed unless knowing K_{VCS} .

- (c) For each voter, KMS-VCS sends RCG the value g^s , where g is a publicly known generator of the ElGamal space. RCG will need these values to assess the correctness of VCS computations during the voting phase.
 - (d) For each voter, using K_{VCS} , the VCS' generates a list B which maps a random identifier to each SSID. This list is passed to the RCG' for the next step, and to the printing service.
5. Establish Party Return Codes (KMS-RCG):
- (a) For each voter, using K_{RCG} , P' and the voter's SSID, KMS-RCG computes the long party codes P'' and the short party codes P''' (return codes to be sent by SMS).
 - (b) A hash of each long party code P'' is mapped to an encryption of its corresponding short party code P''' under key P'' .
 - (c) All mapped pairs in random order per voter are sent to RCG along with information which voter they refer to. (RCG does not learn the return codes P''' , since they are encrypted.)
 - (d) All return codes P''' are associated with the random identifier of list B and sent to the printing service.
 - (e) The storage media of KMS-RCG are destroyed.
6. Establish Position Return Codes (KMS-RCG):
- (a) For each voter, using K_{RCG} , P' and the voter's SSID, KMS-RCG computes the long candidate codes C'' (return codes to be sent by SMS).
 - (b) For each voter and each value T , using K_{RCG} and the voter's SSID KMS-RCG computes the short position codes T''' (return code to be sent by SMS). T is the global value associated with a candidate position in any party list.
 - (c) A hash of each long candidate code C'' is mapped to an encryption of its corresponding short position code T''' under key C'' .
 - (d) All mapped pairs are further processed like the party return codes.
7. Print Voting Cards (Printing Service):
- (a) A first process of the printing service prints the return codes P''' and T''' (received from KMS-RCG) onto each voter's voting card. The voting card is labeled with the random identifier obtained from KMS-RCG.
 - (b) A second process (involving a second printer) uses the list B obtained from KMS-VCS to interpret the random identifier of the voting cards and label them with the home address of the voters. The second process only sees the random identifier of the voting cards. Thus, the printing service should not be able to learn which voters the return codes are associated with.

Note, that after the setup phase no single entity can elicit which voter a return code corresponds with. Furthermore, no single entity can elicit which party or candidate position a return code refers to. Possible attack scenarios require for instance VCS colluding with RCG, information being copied from KMS-RCG and retained prior to the destruction of storage media or the printing service combining the information of both processes.

2.4 Voting

The voting process contains the following steps.

1. The voter accesses the e-voting web-site and a session request is sent to the authentication service. The voter is redirected to MinID and prompted to enter her MinID user name and password. Upon successful authentication on the MinID system, the authentication service sends the voter her voting credentials if she is enlisted in the ERoll. These include her private key s_{Voter} for signing her encrypted vote, the public key of the electoral board h_{EB} for encrypting her vote and the values P and C that represent the competing parties and candidates, mapped to the corresponding party and candidate names.
2. The browser displays the ballot. The voter makes her choice.
3. The voting client application computes an ElGamal encryption $E_{h_{EB}}(\hat{P})$, $\hat{P} \in P$ representing the selected party, and one ElGamal encryption $E_{h_{EB}}(\hat{C}_i)$, per selected candidate $\hat{C}_i \in C$. Further, it signs the encryptions using her private key s_{Voter} and generates one zero-knowledge proof ZKP_1 per encryption to prove that they are not deduced from another voter's submission. The values generated in this step are sent to *VCS*.
4. *VCS* verifies the voter's zero-knowledge proofs ZKP_1 and the signatures to ensure that the vote has been cast by an eligible voter. Upon successful verification, from each encryption it computes the encryptions $E_{h_{RCG}}(\hat{P}^s)$, and $E_{h_{RCG}}(\hat{C}_i^s)$, by partially encrypting with s and partially decrypting with s_{VCS} .⁴ This is easily done knowing the secret value s associated with the voter, knowing the ElGamal private key s_{VCS} , exploiting the relation $s_{VCS} + s_{EB} = s_{RCG}$, as well as the homomorphic property of the ElGamal crypto system. Finally, it computes two zero-knowledge proofs ZKP_2 and ZKP_3 to prove the correctness of its computations. All received values, along with the values generated in this step are sent to *RCG*.⁵
5. *RCG* performs the same verification steps as *VCS*. Additionally it uses the values g^s to verify *VCS*'s computations. Upon success, it uses its private keys s_{RCG} and K_{RCG} to compute the long party codes \hat{P}'' and the long candidate codes \hat{C}_i'' . It computes the hash values to look up the encrypted return codes \hat{P}''' and \hat{T}_i''' in the map established in the setup phase and uses \hat{P}'' and \hat{C}_i'' to decrypt them. If the corresponding entries are found, *RCG* sends the decrypted \hat{P}''' and all \hat{T}_i''' to the voter via SMS. It also sends a signature of approval (voting receipt) to *VCS* to confirm that the vote is accepted and the SMS has been sent. Upon reception, *VCS* permanently stores the vote and forwards the voting receipt to the voter for confirmation. *RCG* saves its signature of approval for the purpose of verification during the tallying phase.
6. The voter compares the codes sent to her by SMS with the values on her voting card.

⁴ *VCS* cannot decrypt the values since it does not know the required private key s_{EB} .

⁵ The obtained encryptions allow *RCG* to obtain the values \hat{P}^s and \hat{C}_i^s . However, *RCG* cannot compute s and thus learns nothing regarding the selected party and the candidates.

2.5 Tallying

The tallying phase contains the following steps.

1. The ERoll and data stored by VCS and RCG during the voting are signed by the corresponding entity and imported in the cleansing service on a physical storage media (e.g. DVD), i.e. the encrypted votes as sent by the voter, her signature, the data stored by RCG, including its signature of approval. The cleansing service thus verifies that the votes recorded in the databases of VCS and RCG are valid and consistently correspond with each other. The latest valid votes of VCS per voter are further processed. Auditors can verify that this process is performed correctly by performing the same steps themselves and comparing their output with the output of the cleansing service.
2. Paper mark-offs from the polling-stations reach the ERoll Service to indicate which voters have cast a paper vote. The digitalized data from the ERoll is imported in the cleansing service from a physical storage device. The cleansing service removes e-votes by voters that cast a paper vote. The cleansing service signs the resulting set of mere votes and transfers them to the offline mix-net. Similarly as above, auditors can verify this process and even import the output into the mix-net themselves on a physical storage media.
3. The first node of the mix net receives the votes from the cleansing service as its input and each node forwards its output to the next one. After the final node has produced its output, the mixed and re-encrypted votes along with the signatures can be accessed by the the auditor through a designated terminal connected to the LAN.
4. The auditor computes and sends a challenge to each mix node. Corresponding with the challenge, each mix-node computes a zero-knowledge proof to prove that it did not alter any votes at mixing. The proofs along with all inputs and outputs of each node data are signed and stored on a DVD to allow verification by the auditor using own equipment. If the signatures and the proofs hold, the auditor imports the DVD in the Decryption / Counting Service.
5. The members of EB provide their shares s_{EB_i} of the private decryption key s_{EB} to the Decryption / Counting Service. The Decryption / Counting Service reconstructs s_{EB} , decrypts the votes and generates a zero-knowledge proof of correct decryption. After the auditor has verified the proof, the votes are counted and the result is published.

3 Transparency

This is the key-measure for the successful application of the subsequent ones. The more information is withheld, the less the public will appreciate the added value gained by applying the remaining measures. As per [27] the idea behind establishing trust among IT-literates is to publish the full logical and technical system documentation - *voting protocol, components, software documents including the source code, all involved parties at development and operation, evaluation reports*

- and relate its analysis to a *security concept*. Trust among the full population will be supported by publishing a *simplified system documentation* that explains and if applicable quantifies the remaining measures for trust establishment. Independent experts who have assessed the full documentation would need to confirm that the simplified documentation has been derived correctly. The public should be informed as early as possible and be able to participate in an open discussion. It should be possible to comment on the project, ask questions and request for more information or clarification.

Transparency in the Norwegian Project. At the point of writing this article, the project is still in its development stage, i.e. the public documentation has not been finalized yet. Although we were able to learn much from the information on the project site, we required further explanations from E-valg and Scytl. Both were very active at providing us with additional detailed information through pre-versions of documents yet to be published, shared documents and personal discussions.

The available documentation shows high quality and is presented in a logical, accessible structure on the Web-site of the Ministry of Local Government and Regional Development. [6] and [2] give an introduction to the project, in [16] technical documents can be found. In [7] the security objectives of the project are stated based on a security domain model and a threat analysis. They formulate technology independent, high level requirements for a secure, transparent and verifiable e-voting system. [21] contains the description and an analysis of the voting protocol that underlies the implemented system. The described mechanism aims at sidestepping single points of failure regarding privacy and the integrity of the ballot. Under [13] the requirements specification, tenders, evaluation and contract documents can be accessed. On the Web site one may also find presentations and videos that document the project.

Documents describing the implemented system were not yet published when writing this article. However pre-versions of the documentations were made available to us without any complicated NDA procedure. In the meantime the source code [12] and system documentation including Common Criteria Security Targets have been published [15]. They explain in [4] that they publish the source code hoping to get useful comments from the public for making improvements. However the auditing software to verify the tallying process is not yet implemented. That will actually be done through a properly open source ("free software") project during the summer.

Recently E-valg have published a couple of more documents in both English and Norwegian. Thus in the short time of preparing this paper, it was hard to get an overview and link these different documents to each other, e.g. which is a refinement of which document or which document has been taken as input for other documents. However, a very valuable document is the one providing a summary of the threat assessment [17]

There is a Web-site to explain the system in a simplified fashion [5]. According to [27] it would be beneficial to additionally relate the explanations to a security concept and underline how and to which degree the security requirements are met.

There is a blog [4] where people from the public can ask questions and place comments on the system. On the same site, the responsables of the project are introduced. E-valg are also active on the social network *twitter* [3]. In order to address concerns from technical experts appropriately, for the future they consider using their issue tracking tool on [12].

4 Separation of Duty

By distributing secrecy-critical duties, one can exclude the event of a single entity being able to break secrecy, i.e. compromise the voters' privacy or elicit partial results prior to the tallying phase. Under separation of duty, secrecy is only broken if a whole group of entities fail (or choose not) to follow their respective procedures correctly.⁶ Since it is effective and easy to explain, [27] captures separation of duty as a measure suited for trust establishment.

Responsibilities can be separated on various levels, i.e. organizational (enforcing restricted access to information within an organization), architectural (physically and logically separating information) and evolutionary (having the organizations in charge use their own equipment, particularly use self-developed or independent 3rd party software). The potential to gain trust heavily relies on the selection of the responsible parties, their ability to perform their duties independently and to confirm to the public that they have done so truthfully.

Separation of Duty in the Norwegian System. Separation of duty will widely be implemented throughout the voting procedure of the Norwegian system. The efforts are summarized as follows.

1. The three respective environments that run RCG, the remaining online components (VCS, authentication service), and the isolated components (KMS, cleansing service, decryption service) are operated by independent governmental departments. Violating secrecy requires that information kept by at least 2 of the 3 sites be shared. The fact that all departments operate at least 100 km apart from each other, induces additional trust in their independence (organizational and architectural separation).
2. The secret key s_{EB} required to decrypt votes is not kept anywhere during the vote casting phase. (It is not even explicitly computed during the setup phase, since its creation is distributed among KMS-VCS and KMS-RCG in the KMS.) Thus, a potentially malicious member of the electoral board EB cannot break the secrecy of the ballot or prematurely elicit partial results even if he gets hold of encrypted votes (e.g. through VCS, RCG, or from malware running on voters' computers).

⁶ Apart from secrecy, separation of duty can also be employed similarly in order to circumvent the violation of a vote's integrity. This aspect is discussed in the context of verifiability in Section [5].

3. The mix-net consists of four nodes. Including the mix-net service in breaking the voter’s privacy thus involves convincing four players (each operator in charge of a mixing node) in participating in an illegal action. Given that the nodes are operated independently from each other, each of their operators can strengthen public trust in privacy just by officially confirming their independent participation.

These precautions are explicitly outlined in the system documentation and therefore likely to have a positive influence at creating trust among the electorate. We believe that their influence will additionally be strengthened by publicly identifying the responsible of each duty, explaining to which degree they are independent (organizational, architectural, evolutionary), and having them confirm to the public that they have acted truthfully.

The following points summarize potential for further enhancements.

1. Key Generation: The key-generation service creates secret keys for all system players. In particular, all information needed to compute the private key s_{EB} used at the decryption of votes is established at KRD. Unfortunately it is inherently difficult for independent auditors to verify that the information is not persistently stored, i.e. that it is not copied before the destruction of the storage media. Having the system players independently compute their own keys can increase trust. Such an enhancement grounds on the mechanism introduced in [23]. Nevertheless, we point out that the secrecy critical keys for VCS and RCG are generated on independent machines of KMS (KMS-VCS and KMS-RCG).
2. VCS / RCG: Since s_{EB} is shared among the members of EB, it is unlikely that any of them can assist any other component at breaking secrecy. However, s_{EB} is also shared among VCS and RCG, due to the relation $s_{VCS} + s_{EB} = s_{RCG}$. Thus, if one of the two players reveals its private key to the other, the latter learns all the information it takes to prematurely decrypt votes and find out who voted how, even without the assistance of KMS. Just as with s_{EB} , trust can be gained by establishing and sharing the keys s_{VCS} and s_{RCG} among multiple entities within the respective organization. Clearly this would yield additional complexity and costs.
3. Mix-Net: Separating the duty of mixing votes among four mixing nodes holds much potential of increasing the public’s confidence. However, the documentation suggests that each node is kept in the same physical environment (KRD). To build trust, it would be beneficial to outline how the independence of the node is enforced, i.e. whether they are meant to be independently maintained, supervised or operated (organizational) and whether they should run independent software (evolutionary). We point out that the E-valg project management and Scytl are in favour of including more independent nodes for mixing at some point.
4. Malicious software running on the voter’s computer, potentially even the voting client application downloaded from the authentication service itself, could forward the voting choices entered by the voter to a third party in

plain-text. E-valg have studied the option of having voters enter personalized codes as their voting choices, i.e. not only have them use codes for verification, but also use codes for expressing their will. However, such an approach has been conciously outruled, due to the inherent loss of usability.

5 Verifiability

In verifiable voting systems voters can verify that their vote is *cast as intended* and *stored as cast* (individual verifiability [22,20]) by accessing the relevant data collected by the voting servers. They can even verify that all collected votes were cast by eligible voters and that all these votes are correctly counted, i.e. one per voter (eligibility and universal verifiability [22,20])⁷ If all processed data can be verified as correct, it becomes obsolete to trust in any system players at preserving the integrity of the vote. Verifiability has been identified as a measure for trust establishment in [27].

Verifiability in the Norwegian System. The SMS received upon casting a vote distinguishes the Norwegian e-voting system from others. It confirms to the voter that her vote has reached VCS as intended. Note, that employing an independent channel for the purpose of verifiability, the event of a corrupted computer is addressed, who potentially may display misleading information to the user (trusted platform problem). In this respect, Evalg do not only comply with point 16 of the Council of Europe in [19], they also exclude the need to trust one's own platform with regard to verification.

On the other hand, the solution comes with a price. Since the voter has no possibility of accessing any public information from the ballot-box, she will inherently need to trust system players regarding the storage of her vote (individual verifiability). Accordingly, she has no means of verifying herself that the tally includes all and none but authorized votes (universal and eligibility verifiability)⁸ Separation of duty thus comes in to play again, this time with respect to verifiability. On the positive side, some separation of duty is actually in place.

5.1 Individual Verifiability

Due to the return codes sent to her by SMS, the voter is able to verify that her first vote has been cast as intended and reached the VCS. She can not verify that VCS actually stores her vote. Nevertheless, this is mitigated by the fact RCG is supposed to store its signature of approval (voting receipt) as pointed out in Section 3. If only one of both refuses its duty, the auditors will observe the inconsistency during the first step of the tallying phase and include that

⁷ Note that in internet voting systems one must weaken the notion of eligibility verifiability. In this article therm captures the ability to verify that all collected votes have been cast *using the credentials* of eligible voters.

⁸ Not offering a public bulletin board is an intentional measure, aiming at circumventing vote buying by individuals from the public.

finding in the report addressed to the authorities of their municipality. Just as in the traditional polling-station elections, the municipality will use the report to decide whether the inconsistency is acceptable or whether they need to fail the election.⁹

When casting subsequent votes, voters have to bring forward some more trust: If voters cast a subsequent vote that contains some repeated choices, the computer can cast a different value for these choices and inform RCG which ones these are. RCG then re-sends the codes as expected by the voter. A similar attack can be performed by a computer that colludes with the device receiving the SMS.

Thus, voters are reassured that their vote is stored as cast when they trust either RCG along with one of the two user devices, or VCS along with the computer. Further, they need to be given at least one report from a trusted auditor that states no inconsistencies.

In the meantime the aim is to enhance the system and have RCG store the votes as well and have a vote counted if at least one out VCS and RCG holds it along with a corresponding voting receipt issued by the other party. Thus, if the voter trusts in the independence of VCS and RCG despite running software from the same vendor, and if she believes in the trustworthiness of at least one of both environments of operation, she can be confident that her vote reaches the tallying stage (despite inconsistencies detected by an auditor). Further, it has been discussed whether to have VCS publish the voting receipts received from RCG. Thus after verifying that the receipt is published, even voters who trust neither VCS nor RCG only need to confide in at least one honest auditor reporting voting receipts that do not correspond with any vote in VCS or RCG. Counterarguments to this approach include usability concerns (how do voters verify that the receipt is published) and the fact that this would yield vote buying by individuals from the public more feasible.

5.2 Universal and Eligibility Verifiability

Universal and eligibility verifiability are not granted to the voter. Nevertheless, the system foresees the auditors to perform verification tasks. In that sense, the voter delegates verification. Clearly, the more trustworthy and the more independent the voter believes an auditor to be from the rest of the system, the more will she trust in the integrity of the final tally. For the sake of public trust, we encourage to employ multiple independent auditors (in a strong sense of separation of duty, i.e. operating on their own equipment and running own software) for verifying the necessary tallying steps and have them confirm to the public that their verification was successful.

⁹ We point out that the auditing service running in Brønnøysund Registry Centre would detect the inconsistency in real-time and allow action prior to the audit. Yet, taking this as the solution would still require full trust in Brønnøysund Registry Centre (since VCS is also run there) and thus offers no mitigation regarding the lack of individual verifiability in terms of *stored as cast* under the organizational and architectural separation of VCS and RCG.

As shown in Section 3, All steps of the tallying phase can be audited - the voter merely needs to trust at least one auditor, i.e. believe that he would publicly reveal failing verification steps. Since anybody can ask to be an auditor and use own equipment to perform verification, the auditability of the system becomes comparable to traditional polling station elections.

6 Further Measures to Establish Trust

In this section we discuss vote updating, standardised security evaluations and test elections as measures to establish trust.

6.1 Vote Updating

According to [27] vote updating increases trust in the Internet voting system for many reasons, e.g. to overcome family voting, vote buying or problems with the PC. However, [27] also mention that it is important to ensure that the 'last' vote is the one that is counted.

Vote Updating in the Norwegian Project. In the Norwegian system vote updating is possible. Voters can repeat the electronic voting procedure several times or cast a paper vote at the polling station, where the latter overrules any electronic vote. This holds in particular if voters do not get the success message on their screen or if they do not get the SMS containing the expected codes.

One point that could be improved is that voters cannot verify whether their electronic vote has been replaced by a paper vote cast by collaborating poll workers in the polling stations. There should be some way for voters to be informed whether their electronic vote has been overwritten by a paper vote. Further, vote-updating protects the system from vote-buying initiated by non-system players. However, vote-buying can still be performed by any entity who sees the vote as encrypted by the voter's computer or the collection of decrypted votes.

6.2 Evaluation

Evaluating the system according to international standards increases trust in the system according to [27]. This is meant to confirm to voters that the developed system corresponds to the one that is documented and that experts analysed it according to widely accepted standards including the formal voting protocol analysis, Common Criteria, ISO 27001, the k -resilience value [25], process observation, and usability standards.

Evaluation in the Norwegian Project. According to the procurement [14], 'The supplier shall in the development process create the necessary documentation for a formal review process and Common Criteria certification to EAL4+¹⁰

¹⁰ EAL means Evaluation Assurance Level; while level 1 is the lowest one and 7 requires the most thorough evaluation.

of all components directly related to e-voting, including counting and returning of members. [...] The supplier shall in the development process of Election System components not directly related to e-voting create the necessary documentation for a Common Criteria certification to EAL2.' It has been decided that for the test run the availability of corresponding documents is sufficient while before using the system after the test election again, the documentation and the system will be undertaken a Common Criteria (CC) evaluation. For average voters these documents are meaningless. While they might know that in general people could now evaluate, they do not know whether a single person/institute has done this.

With respect to CC documentation, the following Security Targets (ST) are available on the Internet:

- Election Administration software according to EAL2 [8]
- Electronic counting of paper votes software according to EAL2 [9]
- Electronic Voting Software according to EAL4+ [10]

These documents seem to be pre-documents yet to be evaluated. [10] does not base on the existing Protection Profile defining basic requirements for Internet voting [28]. One difference is that [10] other than [28] does not include the assumption of trustworthy computers at the voters' side. However, whether this is acceptable by the CC evaluators is questionable, since one might assume that voters will generally not update their votes, which would permit manipulated PC's to elicit who voted how. Other CC documents like the high level system description are also not yet available.

The data centers seem not to have a formal ISO 27001 certificate, However, they are owned by the government and run other critical applications. It is also planned to compute and illustrate the k-resilience value of the whole system according to [25] in order to show more precisely than in section 4 which entities need to be trusted regarding which security property.

A description of the protocol and its analysis exists [21]. However it would be interesting to see an analysis of the implemented system, i.e. without assuming ideal functionality.

Currently there is no information on how the observation of the security critical processes is organized. It is also unclear how the process is defined that ensures that the systems in use for the election correspond to what has been documented and announced on the Web. While we are also not aware of classical user test, the test elections (see section 6.3) can be seen as such. In addition, the reason for using T was based on usability argumentations.

6.3 Test Elections

Test elections were included in the list of trust establishment measures in [27] as it allows voters to experience the full voting process beforehand. Thus, voters' doubts and concerns that emerge from the act of casting their vote itself can be addressed without requiring them to simultaneously question the success of a real election.

Test Elections in the Norwegian Project. It is very hard to describe the technical delta for 10 pre-pilots performed from October of last year up until mid May of this year, and with continuous development in between. However, all 10 pre-pilots have used MinID, and two have in fact used return codes on SMS (in the first test 78% of voters reported checking their code). The last pre-pilot in Re (from May 13-19) used a near-complete system, where all proofs etc. from the protocol were generated.

Using pre-versions of the software may have disadvantages. Voters might be confused about different interfaces and the SMS having not being relevant for the test elections while very important for the legal binding election. However, maybe this is all clearly communicated to those who participated in test elections but we are not aware of this as this information is only available in Norwegian. In principal, one could also define the election in September as test with the full system and all processes implemented.

7 Conclusion

We have described the Norwegian Internet voting system with respect to the measures for trust establishment proposed in [27].

Transparency and the technical measures we discuss imply significant extra costs and complexity for the project. We may conclude that E-valg make significant extra efforts in trust establishment although a high degree of public trust is assumed towards the central election administration. Taken from the published information, as well as from the discussions with the people in charge, it became obvious that being transparent is no easy task even if there is much willingness and even if the legal premises are given. Document management and version control becomes even more important.

Although we identified some possible enhancements to the system, we do not claim by any means that our additional propositions need to be implemented in order for the project to find acceptance among the Norwegian electorate; in particular not for the elections in September. They are rather meant to expose further possibilities that may also be found useful and relevant in the context of Internet voting projects for future large scale deployments as well as in other countries and as a proactive means to address concerns among the public that may arise due to irregularities at operation in the future.

Regarding transparency, one might get the impression that E-valg concentrate very much on the experts while this might also be the case as documents and information for the voters are only available in Norwegian. From our understanding, information regarding the remaining risks, the trust in different entities, and the restrictions of the verifiability are hardly communicated to the public.

With this analysis, we also validated the trust established measures proposed in [27] and demonstrated that it is possible to address all of them in one project.

8 Late Remarks

Since the Norwegian project is still very new, not all information is yet available in the documentation. Also parts of the system naturally tend to change during the initial development process. At least we would like to point out two relevant aspects that we became aware of only at a very late stage of our research.

- RCG accesses MinID to obtain the voters' mobile number in order to send the SMS containing the verification codes. By indicating a wrong mobile number, MinID alone can cast votes without the voters noticing. With regard to individual verifiability, voters will additionally need to trust in MinID not launching any such attacks.
- As described in this paper, the system has been designed and implemented to send voters an SMS containing codes relating to the selected party and the selected candidates, i.e. their position in the party list. However, the SMS's in the fall elections will only contain one code representing the party. Thus, the exposition of verifiability as described in the corresponding chapter only relates to the selected party, not the selected candidates.

References

1. Common Criteria for Information Technology Security Evaluation. Version 3.1, Revision 3, Final (July 2009), <http://www.commoncriteriaportal.org/cc/> (retrieved: June 07, 2011)
2. About the e-vote project @ONLINE (May 2011), <http://www.regjeringen.no/en/dep/krd/prosjekter/e-vote-2011-project.html?id=597658>
3. E-valg 2011 (krd_evalg2011) on twitter @ONLINE (July 2012), <http://www.regjeringen.no/pages/16539918/03SystemArchitecture-Evote.pdf>
4. e-valgbloggen @ONLINE (June 2011), <http://www.e-valgbloggen.no/>
5. E-valglosningen @ONLINE (June 2011), <http://www.regjeringen.no/nb/dep/krd/prosjekter/e-valg-2011-prosjektet/e-valgsystemet1.html?id=597799>
6. E-vote 2011-project @ONLINE (May 2011), <http://www.regjeringen.no/en/dep/krd/prosjekter/e-vote-2011-project.html?id=597658>
7. e-vote 2011 security objectives @ONLINE (May 2011), http://www.regjeringen.no/upload/KRD/Kampanjer/valgportal/e-valg/tekniskdok/Security_Objectives_v2.pdf
8. Election administration software according to eal2 @ONLINE (June 2011), http://www.regjeringen.no/pages/16539918/SecurityTargetforElectionadministrationsoftwarev1_0.pdf
9. Electronic counting of paper votes software according to eal2 @ONLINE (June 2011), http://www.regjeringen.no/pages/16539918/SecurityTargetfore-countingofp-votesv1_0.pdf
10. Electronic voting software according to eal4+ @ONLINE (June 2011), <http://www.regjeringen.no/pages/16539918/SecurityTargetforElectronicVotingSoftware.pdf>
11. Minid @ONLINE (June 2011), <http://minid.difi.no/minid/minid.php?lang=en>
12. source.evalg.stat.no @ONLINE (June 2011), <https://source.evalg.stat.no>

13. Specification, tenders, evaluation and contract @ONLINE (May 2011), http://www.regjeringen.no/upload/KRD/Kampanjer/valgportal/e-valg/tekniskdok/Security_Objectives_v2.pdf
14. System requirements specification @ONLINE (May 2011), http://www.regjeringen.no/upload/KRD/Kampanjer/valgportal/e-valg/Anskaffelse/System_Requirements_Specification1.pdf
15. Systemarkitektur @ONLINE (June 2011), <http://www.regjeringen.no/nb/dep/krd/prosjekter/e-valg-2011-prosjektet/kildekode/dokument.html?id=645240>
16. Technical documents @ONLINE (May 2011), <http://www.regjeringen.no/en/dep/krd/prosjekter/e-vote-2011-project/technical-documents.html?id=612104>
17. Threat assessment summary e-voting, admin, and pvoting toe's @ONLINE (May 2011), <http://www.regjeringen.no/pages/16539918/ThreatAssessmentSummary.pdf>
18. Allepuz, J.P., Castelló, S.G.: Universally verifiable efficient re-encryption mixnet. In: Electronic Voting, pp. 241–254 (2010)
19. Directorate general of democracy and political affairs: Guidelines on transparency of e-enabled elections. GGIS (2010) 5 E, Council of Europe (2010)
20. Gharadaghy, R., Volkamer, M.: Verifiability in electronic voting - explanations for non security expert. In: Krimmer, R., Grimm, R. (eds.) Electronic Voting 2010 - 4th International Conference. LNI, vol. 167, pp. 151–162. Gesellschaft für Informatik, Bonn (2010)
21. Gjøsteen, K.: Analysis of an internet voting protocol. Cryptology ePrint Archive, Report 2010/380 (2010), <http://eprint.iacr.org/>
22. Kremer, S., Ryan, M., Smyth, B.: Election verifiability in electronic voting protocols (2010)
23. 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)
24. Volkamer, M.: Evaluation of Electronic Voting - Requirements and Evaluation Procedures to Support Responsible Election Authorities. LNBIP, vol. 30. Springer (2009)
25. Volkamer, M., Grimm, R.: Determine the Resilience of Evaluated Internet Voting Systems. In: First International Workshop on Requirements Engineering for E-Voting Systems, Atlanta, GA, USA, pp. 47–54. IEEE CS Digital Library (2009), doi:10.1109/RE-VOTE.2009.2
26. Volkamer, M., Schryen, G., Langer, L., Schmidt, A., Buchmann, J.: Elektronische Wahlen: Verifizierung vs. Zertifizierung. In: Fischer, S., Maehle, E., Reischuk, R. (eds.) Informatik 2009: Im Focus das Leben, Beiträge der 39. Jahrestagung der Gesellschaft für Informatik e.V. (GI). LNI, vol. 154, pp. 1827–1836. Gesellschaft für Informatik, Bonn (2009)
27. Volkamer, M., Spycher, O., Dubuis, E.: Measures to establish trust in internet voting. In: ICEGOV. ACM International Conference Proceeding Series. ACM (2011)
28. Volkamer, M., Vogt, R.: Basissatz von Sicherheitsanforderungen an Online-Wahlprodukte (BSI-PP-0037), common Criteria Protection Profile (2008), <http://www.bsi.de/cc/pplist/pplist.html>

Internet Voting System with Cast as Intended Verification

Jordi Puiggalí Allepuz and Sandra Guasch Castelló

Scytl Secure Electronic Voting

Abstract. In remote electronic elections the voting client software is usually in charge of encoding the voting options chosen by the voter. Cast as intended verification methods can be used to audit this process, so that voters do not need to trust the voting client software. In this paper we present the revision of our initial proposal for the eValg2011 project for an Internet voting protocol providing cast as intended verification functionalities, and evaluate its security.

1 Introduction

In remote electronic elections, the voting client software is generally in charge of encoding the voting options chosen by the voter before sending the encrypted vote to a remote voting server. Most of the times, that means that voters have to trust that the voting client is not going to change their selections before they are encrypted. In case the voting client would do it, the probability of being detected is very low. For this purpose, cast as intended verification methods have been designed: voters do not need to trust the voting client software to properly encode the selected voting options, since they can audit the process.

As stated in [1], the purpose of the eValg2011 project is to establish a secure electronic voting solution for general, municipal and county council elections. For this purpose, a set of Objectives of Security [2] for the eVoting system to be implemented for the eValg2011 project was defined for the bidding phase. Specifically, the ability of detecting potential vote manipulations by a malicious voting client when casting a vote was required (zero trust in the voting client).

The aim of this paper is the presentation of our initial proposal for the eValg2011 project for an Internet voting protocol providing cast as intended verification functionalities. The current eValg2011 eVoting system is a modification of this proposal in order to perform some cryptographic operations in the voting server instead of in the voting client. It is not an objective of this paper to compare both proposals but to introduce a revised version of the original proposal. A description of the eValg2011 system currently implemented can be found in [8].

Specifically, we are going to describe the processes of the original proposal that provide cast as intended verifiability of the voting process and evaluate the security of this proposal.

2 Previous Work

The verification done by a voter to check that her vote has been registered correctly in the election platform server is a critical task. This verification must be done in such a way that the voters obtain a verification proof that unequivocally prove if their voting intent has been properly recorded and, at the same time, cannot be used latter to correlate the vote with the voter (in order to prevent vote buying). Another consideration is that the proofs must be verifiable by human means or with assisted means that do not pose any usability and privacy issue. For instance, verification processes that require voters to perform mathematical operations should consider the human limitations.

Several proposals for cast as intended verification in remote electronic voting schemes have been proposed in the literature. In code voting methods, also known as Pollsterless or pre-encrypted ballots [18], [15], [13], [5], [6], [19], [9], [12], the voter receives in advance a voting card with voting codes and validation codes related to the voting options eligible in the election. In order to vote, the voter enters the codes representing her choices in the voting client, which sends them to the voting server. The voting server then replies to the voter with validation codes that are calculated from the received voting codes. The voter can compare them with those assigned to the selected voting options in her voting card, so that she can verify if the encrypted voting options received in the voting server represent her voting intent.

There are other alternative proposals to code voting that do not require the use of voting and validation codes for providing cast as intended verification. This makes the election configuration easier and reduces logistic costs. For instance, in the method proposed in [4], the voter challenges the voting application in order to verify that the encrypted values match her selections: before the vote is cast, the voting application commits to the generated encryption (e.g., showing the digest value of the cipher text) and the voter can opt to challenge the voting application to disclose the contents of the encrypted vote. If the voter requests the verification, the random factor used to encrypt the vote is revealed by this application, so that the voter can verify that the cipher text (according to the commitment generated by the voting application) corresponds to the chosen voting options. After that, the disclosed encrypted vote is dropped and a new encryption with the same selected options is generated again with new random values (in order to prevent vote selling practices). If the voter does not want to verify her encrypted vote, this is sent to the voting service. Since the voting client does not know when the voter will ask for verifying her vote, it is difficult to cheat the voter when encrypting the voting options without being noticed.

When considering which method could be the basis of our proposal, the approach of challenging the voting client was discarded due to usability and independent verification issues: the operations for checking the proper encryption of a vote cannot be done by human means. Since a validation mechanism provided by the voting application could be also modified by the malicious client, the voter needs an external tool or the collaboration of a third party to verify the correct encryption of her vote. Even the election talliers could decrypt the

votes to be audited (at the end of the election) in order to let the voters verify the decrypted contents match their voting intent. However, the process can be complex and difficult to understand for an average voter.

Regarding code voting methods, the approach of comparing the received codes after casting the vote with the validation codes present in a voting card, seems more familiar for the voters (e.g., it is similar to looking if you have won the lottery) and can be done without any support or tool. The drawback, compared with the challenge method, is that the voter needs to trust that the voting cards have been generated properly and their information is not stored anywhere. However, security controls can be implemented to verify that voting cards are correctly generated and there is no disclosure of information.

Furthermore, there are still some usability and accessibility issues remaining in the code voting methods related to the voting process: these methods require voters to enter codes representing the candidates for casting a vote, which could lead to mistakes and is less usable than a click and select interface (which is feasible in challenge methods).

To overcome these issues, our proposal for cast as intended verification introduced a new variant of code voting that does not use codes for casting votes, but keeps the validation codes for allowing voter verification. The novelty of this scheme is that, instead of using voting codes, it uses a click and select scheme for vote casting (in order to make the system more usable) combined with validation codes for voter verifiability.

3 Proposed Solution

3.1 Design Requirements

The main challenge when designing a code voting solution that only uses validation codes, is how to combine probabilistic and deterministic encryption schemes: votes shall be encrypted using a probabilistic encryption scheme in order to maintain voter privacy. However, validation codes are calculated from the encrypted votes and their values have to be always the same (i.e., they should be deterministic) in order to generate the voting cards in advance. Finally, the system shall provide a high level of usability.

3.2 Overview of the Proposed Solution

Voters use validation codes (called return codes in our scheme) for verifying the proper recording of their vote contents in the voting server. To this end, before the voting phase, the voters are equipped with voting cards containing the return codes assigned to each voting option. Voting cards are not linked to a specific voter, so that they may be exchanged between voters or a voter can request more than one card in case she thinks her first card may have been intercepted by an attacker.

The generation of such voting cards is done during the election configuration phase. Some secret keys are involved in the generation of the return codes of

the voting cards, and these keys will be required by the voting platform for generating the return codes from the votes cast. Therefore, during the election configuration stage, these keys are generated, used in the return code generation process and installed in the voting platform. It is assumed that a secure method for protecting the keys is used.

The eVoting platform server side is composed by two main modules: the voting server (Vote Collector Server or VCS), which contains the Ballot Box storing the votes to be counted at the end of the election and participates in the generation of the return codes for voter validation, and the validation server (Return Code Generator or RCG), which generates the final values of the return codes and sends them to the voters.

When the voting phase starts, voters connect to the voting platform, authenticate themselves and select their preferences. After the voter confirms her choices, the voting client encrypts the vote using the election public key.

The voter proceeds to the audit phase in order to verify that her vote is cast as intended. In this phase, she is asked to introduce a special code present in the voting card: the voting card identifier *VCID*. A second encryption is then generated using this value and the RCG public key. The first encryption is the vote which is received in the VCS, stored in the Ballot Box and counted in the tallying process. The second encryption is used by the RCG for generating the return codes associated to the vote contents.

In addition, the voting client generates a cryptographic proof correlating the contents of both encryptions. This prevents a malicious voting client from generating two encryptions based on different voting options in order to cheat the voter (e.g., the encrypted vote over which the return codes are generated contains the selections made by the voter, but the encrypted vote stored in the Ballot Box contains different selections).

Both encryptions and the proof are sent to the remote voting server VCS, which forwards the contents to the RCG.

The RCG verifies the proof and uses its cryptographic keys to generate the return code from the second cipher text. The return code is sent back to the voter using a channel independent from the one used to cast the vote (e.g., SMS).

After that, the RCG notifies the VCS about the successful process sending back a digitally signed hash of the encrypted vote (called the voting receipt). The VCS then stores the first cipher text in the Ballot Box and forwards the voting receipt to the voting client, which shows it to the voter. This receipt can be used as a proof that the vote has been *recorded as cast*: as the VCS publishes the voting receipts corresponding to the contents in the Ballot Box, voters can verify that their votes are correctly stored.

The voter receives the return code and compares it with the code assigned to the selected voting option in her voting card in order to check the correctness of the encrypted vote.

It is assumed that multiple voting is allowed (the system supports it), so that in case wrong return codes are generated the voter can opt to cast her vote from another computer.

If multiple voting is not allowed, the verification of correct encoding of the vote could be done before the casting process: the second encryption is generated in first place and sent to the RCG, which generates the return code and sends it to the voter. In case the verification is successful, the voter can decide to cast the vote (the first encryption), so that it is stored in the Ballot Box.

The ability of asking multiple times for return codes, by multiple voting or by this second system, prevents vote selling practices based on the values of the received return codes.

4 Detailed Cryptographic Protocol

The participants of the voting process are the voter V , the voting client C , the voting server VCS containing the Ballot Box and the validation server RCG, which generates the return codes to be sent to the voter.

The encryption algorithm is ElGamal [7]. During the election configuration phase, the election cryptosystem parameters p , q , g are defined:

- The modulo p is chosen as a large safe prime, where $p = 2q + 1$ and q is a prime number.
- g is a generator of G_q , a q -order subgroup of Z_p^* .

As it has been explained in the overview section, the two encryptions of the vote are calculated under different keys: the election and the RCG public keys.

The election and RCG private keys x_e and x_{rcg} are independently selected from Z_q , and the public keys h_e and h_{rcg} are calculated as $h_e \equiv g^{x_e} \pmod{p}$, $h_{rcg} \equiv g^{x_{rcg}} \pmod{p}$ (modular operations will be obviated from now).

Symmetric keys K_{vcs} and K_{rcg} for the VCS and the RCG respectively are also generated in the configuration phase. The K_{vcs} key will be used to generate the second encryption of the vote, while K_{rcg} will be used to generate the return code values.

The following sections describe the steps of the voting process.

4.1 Vote Preparation for Vote Casting

After the voter confirms her selections, the voting client C proceeds to encrypt the vote using the ElGamal encryption algorithm and the election public key.

The vote is encrypted using a random exponent r in Z_q as:

$$c_{prob} = (v \cdot h_e^r, g^r) = (\alpha, \beta) \quad (1)$$

A proof of knowledge $proof_1$ of the random exponent using the Schnorr Identification Protocol [11] is generated for the encrypted vote in order to prevent reply attacks based on re-using a valid vote from a voter to discern the voter intent based on the return codes received by the attacker.

4.2 Vote Preparation for Verification

A second deterministic encryption using a fixed exponent (in order to be able to generate the return codes) and a cryptographic proof relating both encryptions are generated in the voting client.

This second encryption is based on generating a cipher text using the ElGamal encryption algorithm as a deterministic algorithm by using a fixed exponent generated from information only known by the voter and by the VCS, so that the RCG cannot learn about the voting options selected by the voter when generating the return codes. It is generated in such a way that the resulting cipher text is different for each voter and voting option in order to calculate suitable values for the return codes from it. The cipher text is then re-encrypted using the RCG public key h_{rcg} and the ElGamal encryption algorithm with a random exponent, so that VCS never learns about the selected voting options.

Generation of a Fixed Exponent in the Voting Client. A first fixed exponent is generated in the voting client using information only known by the voter (the voting card identifier $VCID$ and the vote content):

$$a = H(VCID, v), \text{ where } H \text{ is a pseudorandom function [3].}$$

Request of a Second Fixed Exponent to the VCS. The voter enters the $VCID$ code identifying her voting card into the voting client application, which is used to request a value for a fixed exponent to the voting server VCS without revealing it. The VCS uses its secret key K_{vcs} to generate that value:

$$\begin{aligned} C \rightarrow VCS : b &= H(VCID) \\ VCS \rightarrow C : d &= H(K_{vcs}, b), \end{aligned}$$

A blind signature mechanism could be used in order prevent the VCS from knowing the value d .

For example, the VCS could have a keypair t_{vcs} (public), s_{vcs} (private) from an RSA scheme. The Voting Client could then send the value b to the VCS, blinded by a random value r : $b' = r^{t_{vcs}} \cdot b$. The VCS would then answer with $b'' = b'^{s_{vcs}}$, and the Voting Client could recover the exponent $d' = b''/r$.

Encryption with Fixed Exponents Generated by the Voting Client and the VCS. The deterministic encryption of the voting options is generated as:

¹ Since a is part of the fixed exponent used to generate the deterministic encryption of the vote, the function H might be hard to compute in order to prevent effective brute-force attacks to disclose the voter intent. For example, H could be a Password Based Key Derivation Function ([3], [10]), which can be defined as a pseudorandom function slow to compute, and it is usually used to derive cryptographic keys from non-strong passwords. The number of bits generated by the PBKDF shall be defined in such a way that this computation is not too slow to be executed by the voting client while maintaining the robustness of the encryption.

$$c_{det} = (v \cdot h_e^{a+d}, g^{a+d}) = (\alpha', \beta') \quad (2)$$

Generation of Proof of Equality of Plaintexts. The proof ($proof_2$) is used to demonstrate to the RCG that both encryptions, the one which will be stored in the Ballot Box and the one used to generate the return codes, contain the same plaintext (selected voting options). This way, it is ensured that the information which is audited (the contents over which the return codes are calculated) is the same stored in the Ballot Box.

This proof is a Non-Interactive Zero Knowledge Proof based in the Schnorr Signature protocol [17], which is used to prove that both encryptions (α, β) , (α', β') contain the same plaintext (see [14] with corrections in [16]).

In order to do that, both cipher texts are divided and the voting client proves knowledge of the equivalent encryption exponent $(a + d)/r$.

$$(\alpha' / \alpha, \beta' / \beta) = (h_e^{(a+d)/r}, g^{(a+d)/r}) \quad (3)$$

Re-encryption Using the RCG Public Key. The cipher text obtained from the encryption of the vote with fixed exponents is re-encrypted using the El-Gamal encryption algorithm, the RCG public key h_{rcg} and the same random exponent r used for the first encryption. This re-encryption prevents VCS from being able to perform a brute force attack on the value $v \cdot h_e^a$. The RCG will need to remove this re-encryption layer in order to recover α' , which is needed to verify $proof_2$. The value β' is re-encrypted with a different random exponent in order to prevent an observer to infer if the voter has chosen twice the same voting option or not.

$$\alpha'' = \alpha' \cdot h_{rcg}^r \quad (4)$$

$$\beta'' = \beta' \cdot g^{r'} \quad (5)$$

$$c'_{det} = (v \cdot h_e^{a+d} \cdot h_{rcg}^r, g^{a+d} \cdot h_{rcg}^{r'} \cdot g^{r'}) = (\alpha'', \beta'', \gamma) \quad (6)$$

4.3 Vote Casting

Both encryptions and the proofs are digitally signed using the voter credentials and sent to the VCS.

$$C \rightarrow VCS : \text{Sig}((\alpha, \beta), (\alpha'', \beta'', \gamma), proof_1, proof_2; S_{voter})$$

The $proof_1$ and the voter digital signature are verified at reception in the VCS. In case the verification is successful, both encryptions and the proofs are forwarded to the RCG.

$$VCS \rightarrow RCG : \text{Sig}((\alpha, \beta), (\alpha'', \beta'', \gamma), proof_1, proof_2; S_{voter})$$

4.4 Generation of Return Code

The RCG verifies the $proof_1$ and the voter digital signature at reception. If the verification is successful, it performs the following steps in order to generate the return code from the cipher text $(\alpha'', \beta'', \gamma)$.

Decryption with RCG Private Key. The RCG uses its ElGamal private key to decrypt the second cipher text in order to obtain the original deterministic encryption with the fixed exponents generated in the voting client (see Eq. [2](#)):

$$(\alpha', \beta') = (\alpha'' \cdot (\beta^{-x_{rcg}}), \beta'' \cdot (\gamma^{-x_{rcg}})) = (v \cdot h_e^{a+d}, g^{a+d}) \quad (7)$$

Verification of Proof of Equality of Plaintexts. Once the RCG has obtained the cipher text (α', β') , it can verify that this cipher text is a re-encryption of (α, β) using $proof_2$ and following the steps described in [14](#) and [16](#).

Calculation of Return Code. Finally, the RCG generates the return code value using its symmetric key as:

$$RC = H(\alpha'; K_{rcg}) \quad (8)$$

4.5 Delivering Return Code and Voting Receipt

The return code is sent to the voter using an alternative channel to the one used to cast the vote (e.g. SMS). After sending the return code to the voter, a confirmation is sent to the VCS in order to store the encrypted vote in the Ballot Box. This confirmation is the signed hash of the encrypted vote to be stored in the Ballot Box (the voting receipt). Once the VCS has stored the vote, it forwards the voting receipt to the voting client as a confirmation of the recording of the vote. The VCS publishes the voting receipt (hash of the stored vote) in a Bulletin Board.

$$\begin{aligned} RCG &\rightarrow V : RC \\ RCG &\rightarrow VCS : \text{Sig}(H(\alpha, \beta); S_{rcg}) \\ VCS &: \text{Publish Sig}(H(\alpha, \beta); S_{rcg}) \end{aligned}$$

In order to prevent some attacks hidden as transaction problems (e.g., a malicious VCS claims not having received the voting receipt from the RCG, so that the vote is not stored, but the RCG claims having sent that receipt to the VCS), both the VCS and the RCG store the whole set of information generated during the voting phase: the VCS stores the set of information cast by the voter (digitally signed by the voter) and the voting receipt (digitally signed by the RCG). The RCG stores the information cast by the voter and forwarded by the VCS (digitally signed by both the voter and the VCS), the vote encryption with fixed exponent over which the return code is generated (Eq. [2](#)), and the voting receipt.

4.6 Validation of Return Code and Voting Receipt

The voter receives the return code and compares it with the code assigned to the selected voting option in her voting card in order to check the correctness of the encrypted vote.

The voter also receives the voting receipt digitally signed by the RCG, so that she can verify that her vote has been stored in the Ballot Box by comparing this value with the list published by the VCS in the Bulletin Board.

This explanation abstracts a vote as a container of one selected voting option. In case of multiple selections they are independently encrypted and the same process is repeated for each one, so that independent return codes are obtained.

All the information sent from one component to another is digitally signed in order to provide integrity and proof of origin (e.g., it is assumed that encrypted votes are digitally signed by voters).

5 Generation of Voting Cards

Both the voting card identifier $VCID$ and the VCS symmetric key K_{vcs} are very sensitive, since they together can be used to guess the voting options selected by the voter during the voting phase.

Therefore, it is recommended to generate them in two isolated and independent environments. Each environment can be identified as a Voting Card Generation (VCG) module, VCG1 and VCG2. The same environments can then be used to implement a multiparty generation process to calculate the return codes for the voting cards: one environment for generating partial values, and another for generating the final ones.

The generation of partial and final return code values follows similar steps to those defined in the voting protocol. The main difference is that the starting point is not a cipher text, but a cleartext voting option.

Voting Card generation in VCG1

- A random voting card identifier $VCID$ is generated for each set of return codes belonging to a voting card. The length of this $VCID$ must consider both security and usability requirements.
- A fixed exponent is calculated for each voting option using the ballot identifier and a pseudorandom function: $a = H(VCID, v)$.
- A partial return code for each voting option is calculated following the equation:

$$rc' = v \cdot h_e^a \quad (9)$$

- The set of partial return codes generated for each voting card, related to the voting card code VCC , where $VCC = H(VCID)$, is passed to the following module.

Voting Card generation in VCG2

- A second fixed exponent is generated using the VCC value and the VCS symmetric key K_{vcs} as $d = H(K_{vcs}, VCC)$ (see 4.2). In case a blind signature scheme is used, d may be generated as $d = VCC^{s_{vcs}}$.
- The final return code value for each voting option is calculated in two steps, using the fixed exponent and then the RCG symmetric key K_{rcg} :

$$rc'' = rc' \cdot h_e^d, \quad (10)$$

$$RC = H(rc''; K_{rcg}) \quad (11)$$

Two different environments are used to generate the return code values for the voting cards, in such a way that a single entity has no knowledge of the return codes values assigned to a specific voting card. It is assumed that the service in charge of printing the voting cards processes this information in a separate way in order to preserve the secrecy of the voting cards.

6 Security Analysis

The proposal is designed to provide cast as intended and recorded as cast properties, without compromising voter privacy. However, it is important to analyze the robustness of the proposed method against any possible compromise of its components and under which assumptions these components are expected to be deployed to reduce the risks. We are organizing this analysis based on the different stages of the election process and which components could be compromised. For each possible component compromise, the impact, difficulty of implementing an attack and the probability of success are briefly analyzed. It is not the aim of this section to make a strict threat assessment but to provide an initial analysis of the risks.

6.1 Election Configuration

During the election configuration phase the return codes and cryptographic keys needed during the protocol execution are generated. The disclosure of any of these components could potentially compromise the security properties of the proposal.

a) Return codes disclosure: If the return code values are intercepted by a third party at this stage they could be potentially used to compromise voter privacy or to cheat the voter without being noticed, changing the vote to be cast and sending the return codes of the original one. However, these potential attacks cannot be implemented without compromising other components of the system.

Voter privacy only can be compromised if the attacker also intercepts the return codes of the selected voting options when the vote is cast (i.e., intercepts the communications in the alternative channel or gains access to the RCG). Furthermore, voting cards are not linked to voters, can be exchanged between voters, and one return code value is probable to be connected to different options in different voting cards. Therefore, an attacker also needs to gain access to the voting card identifier to make the attack effective. Nevertheless, if multiple voting is allowed, voters can cast a vote latter on using a different voting machine and a different voting card. Therefore, the attacker needs to intercept all these communications as well.

For cheating the voter, the malicious voting client needs to send the manipulated vote to the VCS and the real selections made by the voter to the RCG. That way, the RCG could send the return codes of the options selected by the voter instead of the options contained in the vote. Otherwise, the voter could detect the attack when receiving the SMS messages with return codes of the manipulated vote. The protocol does not require any direct connection between the client and the RCG and therefore, this should be explicitly done without being detected (i.e., requires also control of the RCG server infrastructure). The success of the attack will depend also on the ability to the voter for casting another vote in another machine. If so, the attacker needs also to control any machine that could be used by the voter. The scalability of this attack depends also in the number of machines controlled, the number of voting cards stolen and the ability to control the RCG and hide unexpected connections from Internet to this machine (initially it should not accept any connection from Internet).

As a general assumption, it is expected that the return code generation process is implemented using the two-step process mentioned in Section 5, using an isolated environment under the supervision of auditors. The files with the return codes to be printed should be exported in two different files encrypted using two private keys, so that two separate services process them independently. Voting cards will be printed in tamper evident sealed blind envelopes (e.g., PIN envelopes). The storage devices used in this process are destroyed or completely wiped as soon as they are not longer required. This prevents accidental disclosure of information that could potentially be used in other attacks. At this stage of the voting process, the main risk is compromising the whole set of voting cards, since this could allow a large scale attack. Compromising only one or a small set makes attacks more difficult to target to a specific voter or reduces their impact.

b) VCS K_{vcs} symmetric key disclosure: in case this key is compromised, the attacker could try a brute force attack focused to disclose the voter intent from the deterministic encrypted vote (see Eq. 9), if it also colludes with the RCG. If so, the hardness of the attack will depend on the length of the voting card identifier $VCID$ (e.g., for a 16 char base32 identifier, it will be 80bits). The use of a slow function as a PBKDF to generate the fixed exponent in Eq. 9 would increase the time needed to disclose the vote contents with such a brute force attack.

As a general assumption, this K_{vcs} key is generated using a cryptographically secure PRNG in an isolated environment (e.g., HSM) and it has a length of at least 128 bits. The length of the voting card identifier $VCID$ should be of at least 80 bits and it should be generated using also a cryptographically secure PRNG.

c) VCS Signature key disclosure: assuming that the RCG also stores the vote, if this key is compromised, an attacker colluding with the RCG could store in this machine a vote that has not been stored in the VCS but with a proof that says it should (the digital signature of a vote stored in the RCG using the VCS private key). The main motivation of this attack is to disrupt the election and try to discredit the VCS, since privacy of the voter is not compromised. This attack also requires having access to a vote cast by a valid voter but that has not been stored in the VCS (since the voter has to be digitally signed using voter credentials). The attack could be detected if the contents stored in the RCG are crosschecked against the voting receipts published by the VCS in the Bulletin Board.

As a general assumption, this key is generated using a cryptographically secure PRNG in an isolated environment (e.g., HSM) and it has a length of at least 2048 bits. It is also expected that the contents of the RCG and the Bulletin board are checked periodically to detect discrepancies.

d) RCG x_{rcg} ElGamal private key disclosure: if this key is compromised, the attacker can decrypt the second encryption cast by the voter (see Eq. 2) in order to recover the vote encrypted by the fixed exponents. However, this information does not disclose enough details to discern the voter intent. Only colluding with the VCS the attacker could try a brute force attack over the voting card identifier $VCID$.

As a general assumption, this key is generated using a cryptographically secure PRNG in an isolated environment (e.g., HSM) and it has a length of at least 2048 bits. The length of the voting card identifier should be at least of 80 bits and it should be generated using also a cryptographically secure PRNG.

e) RCG Signature key disclosure: if this key is compromised and the attacker is colluding with the VCS, she could sign the voting receipt of a vote in order to provide confirmation to the voting applet without forwarding the vote the RCG. In this case, if the vote is not stored in the Ballot Box, this could be detected, since the voting receipt is not published in the Bulletin Board. Publishing the voting receipt without storing the vote can be detected crosschecking both repositories. In any case, the voter could detect this attack since she does not receive the return codes of the cast vote.

As a general assumption, this key is generated using a cryptographically secure PRNG in an isolated environment (e.g., HSM) and it has a length of at least 2048 bits. It is also expected that the contents of the Ballot Box and the bulletin board are checked periodically to detect discrepancies.

f) Election x_e ElGamal private key disclosure: if this key is compromised and attacker could decrypt any vote and compromise voter privacy.

As a general assumption, this key is generated using a cryptographically secure PRNG in an isolated environment and split in shares using a secure multiparty computation protocol with a pre-defined threshold. Shares shall be distributed among members of an Electoral Board with divergent interest, using a HSM (smartcards). The system shall not store any copy of the shares or the whole election private key. All the storage devices (except the smartcards of the electoral board members) shall be wiped or destroyed. The key shall have a length of at least 2048 bits.

6.2 Voting Process

During the voting phase, attacks can be targeted to the voters that are participating (or not) in the election, and against the votes cast. Assuming that attackers did not succeed on compromising the components in the configuration phase, we will consider in this stage risks in case of component compromise during the voting phase. Risks based on a previous compromise are the same as described before. In this phase, we will consider the risks more related to a system component compromise (Voting Client, VCS and RCG) rather than a cryptographic component compromise (keys). The only exception is the set of return codes (voting cards).

a) Return codes disclosure: In this phase, compromising a meaningful number of voting cards is more complex since they are already in possession of voters and the information of the configuration phase has been destroyed (i.e., it will require physical access to each voter location). Therefore, the attacks can be mainly focused on coercion or vote buying practices. Since voting cards are not linked to voters, if voters are allowed to cast more than one vote using different voting cards, coercers or vote buyers cannot use the voting cards to be sure of the voter intent even though they keep these voting cards.

As a general assumption, it is expected that the return code generation process destroys all the information of the return codes after voting cards are printed. It is also an assumption that voters are allowed to vote multiple times. Since the vote casting and vote verification processes can be executed in different order, in case the voter only can cast one vote, the processes could be reversed, sending first the deterministic encrypted vote to check the return codes before casting the vote. This is an option whose impact we are still evaluating in case the voter does not cast the vote, makes changes in the selection and requests again the return codes (sending another deterministic vote).

b) Voting client compromise: If the voting client is compromised (virus, malware, etc.), any attempt to make changes in the voter intent will be detected by the voter when checking the return codes. RCG can also detect any attempt of using a deterministic encryption with a different content than the vote (by

means of a cryptographic proof, see Section 4.2). Only collusion between the voting client and the RCG could pose some risk, as described in the section a) of the Risks during the Election configuration process. However, this will require also compromise other components of the infrastructure to hide any direct dialog between the voting client and the RCG.

As a general assumption, it is expected that the RCG is completely isolated from the public network (i.e., Internet) and connections restricted to those coming from the VCS. Voting client is expected to be digitally signed to prevent the use of manipulated or non-validated client software.

c) VCS compromise: If the VCS is compromised, it cannot gain access to the voter intent (it cannot decrypt the information) or manipulate the vote, since it is assumed that is digitally signed by the voter. It could disrupt the election by not storing the votes but publishing the voting receipts (to prevent voters to detect that their votes were not recorded as cast). However, this malicious practice could be detected by periodically cross-checking the Ballot Box contents with the Bulletin Board ones. If the RCG is also storing a copy of the valid votes from which it has generated return codes, votes eliminated by the VCS could be recovered. Only collusion between VCS and RCG could prevent the recovery of the vote. Attacks described in section b) and c) also apply in this case.

As a general assumption, it is expected that the keys of the VCS are stored in a cryptographic device that prevents them from being exported (HSM). Periodical checks of the Ballot Box and Bulletin Board contents are also expected.

d) RCG compromise: If the RCG is compromised, it can send return codes to voters without sending the voting receipt to the VCS, so that the votes are not stored in the Ballot Box. However, this will be detected by voters when a timeout is reached in their voting client since the receipt is not received. In the other hand, trying to guess the selected voting options after decrypting the deterministic vote (see Eq. 7) is not feasible. Only with the collaboration of the VCS, that could recover its part of the fixed exponent, could reduce the complexity of the brute force attack to the length of the voting card identifier value (usually 80 bits). The other attacks described in sections d) and e) of the Election configuration phase also apply in this case.

As a general assumption, it is expected that the keys of the RCG are stored in a cryptographic device that prevents them from being exported (HSM). Periodical checks of the RCG stored information (validated votes and voting receipts) and Bulletin Board contents are also expected.

6.3 Post Election

Once the election is done, the main risks are related to threat the data that has been registered during the election. Mainly, the following datasets will be vulnerable to attacks: the votes encrypted under the election public key, the votes encrypted under the RCG public key, votes decrypted with the RCG private

key and the voting receipts. The attack objectives in all the cases could be to compromise voter privacy or to disrupt the election.

As a general assumption, it is expected that the encryption and digital signature keys of the VCS and RCG, as well as the electoral board shares are destroyed once they are no longer required (usually when the results are obtained). It is also expected that vote encryption sets are not public disclosed. Only the Bulletin Board contents (voting receipts) and the decrypted votes will be public available.

a) Attacks on the votes encrypted under the election public key or the RCG public key: Crypto-analysis attacks for disclosing the encrypted voter intent are not expected in these votes. In fact it is expected that these attacks will be more probably studied in the votes decrypted by the RCG for the nature of the deterministic encryption exponent. In the other hand, any attempt to manipulate these votes to disrupt the election could be detected by checking the digital signatures and the ZKP that connect them to the deterministically encrypted votes. Therefore, no successful attacks are expected.

b) Attacks on the votes decrypted with the RCG private key: Since these votes derivate the encryption exponent in a deterministic way, it is expected that attacks will be concentrated on breaking this key derivation mechanism. Three components are used for the exponent derivation: the voting card identifier, the selected voting option and a VCS key. Considering that the set of voting options is very limited, we can consider that the strength will depend on the voting card identifier and the VCS key strength. The VCS key will have 2048 bits length and, therefore, is the main one that provides long term protection. If this key is compromised, the strength will depend on the length of the voting card identifier that usually is limited for usability issues since the voters need to type it (e.g., 80 bits). This will make an attack more feasible in a shorter time-frame, since at the end the fixed exponent protecting the privacy of the voting options in the RCG is derived from a master key with lower entropy. Solutions to increase this strength without compromising usability are under examination for future improvements. It is also important to consider that if the voting card identifiers are known in advance, the attack could be straight forward. Therefore, preserving the *VCID*'s as secret and destroying them at the end of the election is of paramount importance to prevent any possible success of this attack.

c) Voting Receipt attacks: Attacks to voting receipts could be focused to discern the original cipher text cast by the voters (since initially these encrypted votes are not public available). However, this will be practically unfeasible since the receipt value is obtained using a hash function.

6.4 Summary

The system is not vulnerable to individual component compromise, requiring in most cases the collusion of several components for starting to design an attack. The main risks are located in the election configuration period, where the voting cards and keys could be compromised. However, these keys can be created in isolated environments (i.e., do not require online processes) and the voting cards can be created using incremental steps in different isolated environments that will prevent having access to all the information at the same time.

The second important risk is related to the protection of the VCS symmetric key. Since this key is required to compensate the strength limitations of the voting card identifier (for usability reasons), it is important to it keep safe and destroy it when the election is done. Otherwise, crypto-analysis attacks could be designed that could make possible an attack in an acceptable timeframe (not confirmed yet). Further work is done in this area.

Other attacks require complex collusions that are more difficult to implement without being detected. Furthermore, the impact in case of success seems more limited to disrupt the process.

References

1. <http://www.regjeringen.no/en/dep/krd/prosjekter/e-vote-2011-project/about-the-e-vote-project/presentation-of-the-project.html?id=598565>
2. evalg 2011 security objectives, http://www.regjeringen.no/upload/KRD/Kampanjer/valgportal/e-valg/tekniskdok/Security_Objectives_v2.pdf
3. Pkcs#5, note = <http://www.rsa.com/rsalabs/node.asp?id=2127>
4. Adida, B.: Helios: Web-based open-audit voting. In: van Oorschot, P.C. (ed.) USENIX Security Symposium, pp. 335–348. USENIX Association (2008)
5. Chaum, D.: Surevote: Technical overview. In: Proceedings of the Workshop on Trustworthy Elections, WOTE 2001 (2001)
6. CESG (Communications and Electronic Security Group). E-voting security study, annex C (2002), <http://www.edemocracy.gov.uk/library/papers/study.pdf2002>
7. El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory 31(4) (1985)
8. Gjøsteen, K.: Analysis of an internet voting protocol. Cryptology ePrint Archive, Report 2010/380 (2010), <http://eprint.iacr.org/>
9. 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)
10. American National Standards Institute. Accredited standards committee x9 working draft. ANSI X9.42-1993: Public Key Cryptography for the Financial Services Industry: Management of Symmetric Algorithm Keys Using Diffie-Hellman (1994)
11. Jakobsson, M.: A Practical Mix. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 448–461. Springer, Heidelberg (1998)

12. 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. Dagstuhl Seminar Proceedings*, vol. 07311. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany (2007)
13. Malkhi, D., Margo, O., Pavlov, E.: E-voting without 'Cryptography'. In: Blaze, M. (ed.) *FC 2002. LNCS*, vol. 2357, pp. 1–15. Springer, Heidelberg (2003)
14. Markus, J., Ari, J.: Millimix: Mixing in small batches. Technical report (1999)
15. Morales-Rocha, V., Soriano, M., Puiggali, J.: New voter verification scheme using pre-encrypted ballots. *Computer Communications* 32(7-10), 1219–1227 (2009)
16. Nguyen, L., Safavi-Naini, R.: Breaking and Mending Resilient Mix-Nets. In: Dingledine, R. (ed.) *PET 2003. LNCS*, vol. 2760, pp. 66–80. Springer, Heidelberg (2003)
17. Schnorr, C.-P.: Efficient signature generation by smart cards. *J. Cryptology* 4(3), 161–174 (1991)
18. Storer, T.W.: Practical pollsterless remote electronic voting (2006)
19. Voutsis, N., Zimmermann, F.: Anonymous code lists for secure electronic voting over insecure mobile channels. In: *Proceedings of Euro mGov*. Sussex University, Brighton (2005)

Linear Logical Voting Protocols

Henry DeYoung¹ and Carsten Schürmann²

¹ Carnegie Mellon University, Pittsburgh, PA, USA
hdeyoung@cs.cmu.edu

² IT University of Copenhagen, Copenhagen, Denmark
carsten@itu.dk

Abstract. Current approaches to electronic implementations of voting protocols involve translating legal text to source code of an imperative programming language. Because the gap between legal text and source code is very large, it is difficult to trust that the program meets its legal specification. In response, we promote linear logic as a high-level language for both specifying and implementing voting protocols. Our linear logical specifications of the single-winner first-past-the-post (SW-FPTP) and single transferable vote (STV) protocols demonstrate that this approach leads to concise implementations that closely correspond to their legal specification, thereby increasing trust.

1 Introduction

Determining the outcome of an election is rarely as straightforward as simply counting the votes and declaring the candidate with the most votes to be the winner. Even for relatively simple voting protocols, such as first-past-the-post, election laws prescribe the detailed provisions for tallying votes and computing the final result. Legal language is precise enough to be used in courts of law to settle debates about the lawfulness of a traditional election implementation (e.g., one that uses paper ballots), but computer-based implementations pose unique challenges.

Because election laws are not written in a formal language, they cannot be directly executed by a computer. Instead, humans translate the legal text to source code of a programming language; typically a general-purpose imperative language, such as Java or C, is the target.

However, this approach to computer-based implementations is problematic: it is unreasonable to expect that the translation from the informal specification in legal text to its implementation as source code will be trusted outright. In large part, this is because programs written in general-purpose imperative languages are comparatively low-level and complex. To verify that such programs correctly implement their legal specifications, one must reason about concrete data structures, exotic language features (e.g., inheritance and method overloading), and vast third-party libraries. Certifying all of these components in full detail is extremely challenging and costly, if not impossible—the gap between legal text and source code is simply much too large.

In response to these problems, this paper proposes to use formal logic—more specifically, *linear logic*—as a foundation for electronic elections. First, logic will serve as an intermediate, formal specification language: rather than translating the legal text to low-level source code, it will be translated to a set of logical formulas. Because the logic allows a high level of abstraction, these formulas will be in close correspondence with the legal text from which they were derived. This minimizes the conceptual distance between the two, thereby increasing trustworthiness.

Second, by way of logic programming languages like the well-known Prolog [6], formal logic also provides a means of programming voting protocols declaratively. The declarative programming paradigm has the advantage of narrowing the gap between the formal specification and its implementation. In logic programming, for instance, the specification’s logical formulas and the program’s source code are one and the same; the operational behavior of the program is derived from a fixed proof-search strategy for the logical connectives that comprise the specification.

Thus, the trusted components of this approach are: 1) that the formal, logical specification adequately reflects the informal, legal specification; and 2) that the logic programming engine is correctly implemented. Requiring some degree of trust in the adequacy of the logical specification is unavoidable; however, as described above, because the logical specification is in close correspondence with the legal text, the conceptual gap to be bridged by adequacy is minimized and trustworthiness increases. Moreover, instead of trusting the logic programming engine, one can choose to trust a much simpler proof checker that validates the explicit proof objects produced by the engine; these proof objects are essentially human-readable abstract execution traces, and therefore can also be audited.

To meet the above goals, first-order logic (and its corresponding logic programming language, Prolog) would indeed be a technically adequate choice of logic. But, we contend that it is not an ideal choice. This paper instead advocates the use of linear logic [12,14], a logic in which assumptions are treated as resources that must be used exactly once. (Sect. 2 provides a brief introduction to linear logic and contrasts it with first-order logic.)

To illustrate the benefits of linear logic as a foundation for electronic elections, this paper presents full linear logical specifications of two voting protocols: single-winner first-past-the-post (SW-FPTP, Sect. 3) and proportional representation through the single transferable vote (STV, Sect. 5). Both protocols are widely used in practice: for example, first-past-the-post (also known as winner-take-all) for national elections in the United States and single transferable vote for parliamentary elections in Ireland, Malta, and Australia. More importantly, these protocols are valuable benchmarks because they represent two extremes of protocol complexity. That elegant characterizations of two such diverse protocols are possible speaks to linear logic’s robustness as a specification language.

Linear logical specifications of these protocols can be transliterated, in a fully syntactic way, to source code of linear logic programming languages, such as LolliMon [13] and Celf [14]. To provide intuition about the operational behavior of

specifications, Sect. 4 of this paper sketches a typical Celf execution of the single-winner first-past-the-post protocol. The transliterations of both protocols to Celf syntax are available at <http://www.itu.dk/~carsten/files/voteid2011.tgz>, if the reader wishes to experiment further.

Another benefit of our linear logical approach is the ability to formally prove the operational correctness of specifications using existing techniques [10,9]. In Sect. 6, we sketch proofs of correctness for the SW-FPTP and STV protocols.

Finally, it is also important to clarify what this paper does not set out to accomplish. First, this paper concentrates on *verified* elections, wherein an a priori static analysis verifies that the election software meets its specification. This is in contrast with *voter-verifiable* elections, which use end-to-end techniques, such as Prêt à Voter [5], whereby voters can convince themselves that the final tally is correct. In general, we believe that the two approaches are complementary: even with end-to-end techniques for detecting anomalies, one should still strive to minimize the occurrence of such costly errors beforehand by running verified software.

Second, beyond operational correctness, voters expect voting protocols to possess security properties such as privacy and coercion-resistance. Although we contend that linear logical protocol specifications will be readily amenable to reasoning about such meta-theoretic properties, we leave this to future work.

Related Work. In the past, there have been attempts to formally prove the implementation of an electronic voting protocol correct. Using the applied π -calculus, Delaune, Kremer, and Ryan [8] modeled a simple protocol and proved its fairness, eligibility, privacy, receipt-freeness, and coercion-resistance with Blanchet’s ProVerif tool. Their work differs from ours in that they concentrate on the security of the protocol whereas we are interested in the auditable correctness of implementations. Perhaps most closely related is the work by Cochran and Kiniry on specifying STV in JML and ESC/Java [7]. Unfortunately, JML and ESC/Java are logically unsound. Program verification must be supplemented by testing to guarantee a reasonable level of program correctness. In contrast, our logical approach to implementing STV guarantees correctness automatically, thereby rendering testing superfluous. Aside from implementation concerns, the literature also contains proposals to improve the security of the STV protocol using cryptography [2]. Those ideas are largely orthogonal to the ambitions of this paper.

2 A Brief Introduction to Linear Logic¹

Traditional first-order logic is concerned solely with truth. Being an abstract idea, truth is inherently free. Consequently, in traditional logic, each logical assumption may be used as many or as few times (including none) as desired—it has no cost.

¹ We encourage the reader who is interested in a more complete introduction to linear logic to refer to Philip Wadler’s excellent tutorial [15].

On the other hand, linear logic admits that, unlike truth, not everything is free. It instead concerns itself with consumable, valuable resources. Because resources are consumable, they may not be freely duplicated and may be used at most once; because resources are also valuable, they may not be freely disposed and must be used at least once. To reflect this, linear logic represents resources as logical assumptions that must be used *exactly* once. With this *resource discipline*, linear logic is able to express—more elegantly and concisely than can traditional first-order logic—operations that must occur only once. Because voting protocols in particular rely significantly on the one-occurrence idiom (e.g., registering each voter only once or counting each ballot only once), this elegance is crucial to minimizing the conceptual gap between the informal, legal specification and the formal, logical specification.

2.1 Connectives of Linear Logic

Full linear logic contains a rich set of connectives for building formulas. To assign a logic programming interpretation, restrictions must be placed on the ways in which these connectives fit together so that proof search becomes more deterministic. Thus, in Celf [14] and its predecessor, LolliMon [13], the formulas of linear logic are polarized into positive and negative classes [1] and a monad is used to prevent interference between the two [16]. For the examples in this paper, only the following fragment of polarized monadic linear logic is needed:

$$\begin{array}{ll} \text{Negative Formulas} & A^-, B^- ::= P^- \mid \forall x:\tau. A^- \mid A^+ \multimap \{B^+\} \\ \text{Positive Formulas} & A^+, B^+ ::= A^+ \otimes B^+ \mid \mathbf{1} \mid !A^- \mid A^- \end{array}$$

The fragment includes atomic formulas P^- , universal quantification $\forall x:\tau. A^-$, linear implication $A^+ \multimap \{B^+\}$, simultaneous conjunction $A^+ \otimes B^+$ and its unit $\mathbf{1}$, the unrestricted modality $!A^-$, and an inclusion, A^- , of negative formulas as positive formulas.

To present the meanings of these connectives, we will now develop a specification of voter check-in at a polling place. Prior to election day, each voter receives a voting authorization card in the mail. To check in at her designated polling place on election day, the voter exchanges her voting authorization card for a blank ballot form. Because each voter receives only one authorization card, the card thus helps prevent ballot stuffing.

In traditional logic, one might try to specify this check-in process by taking as an axiom the formula

$$\textit{voting-auth-card} \rightarrow \textit{blank-ballot} \quad :$$

if a voter has a voting authorization card, then she may have a blank ballot form. However, this specification would allow proofs of such nonsense as $\textit{voting-auth-card} \rightarrow \textit{blank-ballot} \wedge \textit{voting-auth-card}$: if a voter has a voting authorization card, she can receive a blank ballot and keep her authorization card. By iterating this proof, one can show that, under this specification, ballot stuffing is possible: $\textit{voting-auth-card} \rightarrow \textit{blank-ballot} \wedge \dots \wedge \textit{blank-ballot} \wedge \textit{voting-auth-card}$. Therefore, this specification of the check-in procedure is clearly unsound.

Linear Implication, \multimap . The problem is one of expressivity—traditional implication does not express that the check-in process *consumes* the voter’s authorization card. But, as a logic of resources, linear logic provides just the right expressive power. It includes the linear implication formula $A^+ \multimap \{B^+\}$, which, like the traditional implication, is a procedure for producing resource B^+ if given A^+ ; unlike the traditional implication, however, this procedure consumes resource A^+ as part of the production.²

Thus, a sound specification of voter check-in is given by taking as an axiom the linear logical formula

$$\textit{voting-auth-card} \multimap \{\textit{blank-ballot}\} \quad :$$

the check-in process consumes the voter’s authorization card and gives her a blank ballot in exchange.

Simultaneous Conjunction, \otimes , and Its Unit, $\mathbf{1}$. Now suppose that voters are also required to present a photo ID during check-in. The specification will have the same basic structure: ‘a voting authorization card and a photo ID’ $\multimap \{\textit{blank-ballot}\}$. But how can we express the ‘a voting authorization card *and* a photo ID’ resource as a formula of linear logic?

Fortunately, linear logic provides a simultaneous conjunction, $A^+ \otimes B^+$ (read ‘both resources A^+ and B^+ ’). Thus, a specification of the revised check-in process can be given by the formula

$$\textit{voting-auth-card} \otimes \textit{photo-ID} \multimap \{\textit{blank-ballot}\} \quad :$$

when a voter gives a voting authorization card and a photo ID, she receives a blank ballot form in exchange. (Note that \otimes binds more tightly than \multimap .)

Linear logic also includes a unit for simultaneous conjunction, $\mathbf{1}$ (read ‘nothing’), which represents the empty collection of resources. The proposition $\mathbf{1}$ is primarily used in the idiom $A^+ \multimap \{\mathbf{1}\}$, which consumes resource A^+ and produces nothing in return.

Unrestricted Modality, $!$. The prior specification of the check-in process, $\textit{voting-auth-card} \otimes \textit{photo-ID} \multimap \{\textit{blank-ballot}\}$, is not fully satisfactory, however. Because *photo-ID* is treated as a resource and linear implication (which consumes the resources it is given) is used, this axiom specifies a check-in process in which voters must relinquish their photo IDs to vote! This is not the intent; voters should always retain their photo IDs. And so, at first glance, *photo-ID* does not appear to fit into the resource discipline of linear logic.

However, the unrestricted modality, $!A^-$, of linear logic provides a way out. The proposition $!A^-$ is a version of A^- that is not subject to the resource

² The braces around B^+ denote a monad that is not found in conventional presentations of linear logic. It is used to give a committed-choice operational semantics for the logic programming interpretation [13] that is important to our work.

discipline—an assumption $!A^-$ can be used an unlimited number of times (including none). Alternatively, one may think of $!A^-$ as stating that A^- is a fact that will remain true regardless of how the system evolves.

Using the $!$ modality, the revised specification can therefore be given by

$$\textit{voting-auth-card} \otimes !\textit{photo-ID} \multimap \{\textit{blank-ballot}\} \quad :$$

when a voter gives an authorization card and shows a photo ID, she receives a blank ballot form. (Note that $!$ binds more tightly than \otimes and \multimap .)

Universal Quantification, $\forall x:\tau$. Strictly speaking, the current specification of voter check-in does not capture the requirement that the name on the authorization card must match the name on the photo ID.

This problem can be resolved using universal quantification. In linear logic, multi-sorted universal quantification, $\forall x:\tau. A^-$, behaves just as in traditional logic. In particular, the members of the domain of quantification are not subject to a resource discipline. Thus, the specification may be revised to

$$\forall v:\textit{voter}. (\textit{voting-auth-card}(v) \otimes !\textit{photo-ID}(v) \multimap \{\textit{blank-ballot}\}) \quad :$$

when a voter v gives *her* authorization card and shows *her* photo ID, she receives a blank ballot form.

3 A Linear Logical Specification of First-Past-the-Post

To demonstrate how linear logic can be used to specify voting systems, we now present a concise, elegant specification of single-winner first-past-the-post voting (SW-FPTP). In SW-FPTP voting, each voter casts a ballot on which she has selected a single candidate. After all ballots have been counted, the candidate with greatest vote total is determined; this candidate is declared the winner. Because SW-FPTP voting is relatively simple, it makes an ideal first example.

For our specification of SW-FPTP, we must introduce several predicates, which are summarized in Table II. The *uncounted-ballot*, *hopeful*, *defeated*, and *elected* predicates are used to characterize the ballot box and the candidates' electoral statuses, and the *count-ballots* and *determine-max* predicates indicate progress through the algorithm's two phases. We also assume the existence of the usual ordering predicates on natural numbers, such as $!(N \geq N')$.

SW-FPTP voting is specified by the collection of linear logical axioms shown in Fig. III. (For conciseness, we follow the standard convention that universal quantification is implicit for all variables written in upper case.) The *count/run* and *count/done* axioms specify how the ballot counting phase of SW-FPTP works, whereas *max/run* and *max/done* characterize a random tournament for finding the candidate who has the greatest vote total. Although it would be straightforward to use other tie-breaking criteria, for simplicity of presentation we will assume that ties are broken arbitrarily.

Table 1. Descriptions of predicates used in the SW-FPTP specification

Predicate	Meaning
$uncounted-ballot(C)$	An uncounted ballot for candidate C .
$hopeful(C, N)$	Candidate C is not yet defeated nor elected, and N ballots have been counted for C thus far.
$!defeated(C)$	Candidate C has been (and will remain) defeated.
$!elected(C)$	Candidate C has been (and will remain) elected.
$count-ballots(U, H)$	Token to indicate that the algorithm is in the process of counting ballots; there are U uncounted ballots remaining, and H candidates are hopefuls.
$determine-max(H)$	Token to indicate that the algorithm is in the process of determining which candidate has the greatest vote total; there are H hopeful candidates remaining.

$$\begin{aligned}
 \text{count/run} &: \text{count-ballots}(U, H) \otimes \\
 &\quad \text{uncounted-ballot}(C) \otimes \text{hopeful}(C, N) \\
 &\quad \multimap \{ \text{hopeful}(C, N+1) \otimes \text{count-ballots}(U-1, H) \} \\
 \text{count/done} &: \text{count-ballots}(0, H) \\
 &\quad \multimap \{ \text{determine-max}(H) \} \\
 \text{max/run} &: \text{determine-max}(H) \otimes \\
 &\quad \text{hopeful}(C, N) \otimes \text{hopeful}(C', N') \otimes !(N \geq N') \\
 &\quad \multimap \{ \text{hopeful}(C, N) \otimes !\text{defeated}(C') \otimes \text{determine-max}(H-1) \} \\
 \text{max/done} &: \text{determine-max}(1) \otimes \text{hopeful}(C, N) \\
 &\quad \multimap \{ !\text{elected}(C) \}
 \end{aligned}$$

Fig. 1. A linear logical specification of single-winner first-past-the-post voting

3.1 Counting Ballots with Count/Run and Count/Done

An intuitive reading of the axioms used to specify SW-FPTP ballot counting is:

count/run: ‘If we are in the process of counting ballots ($count-ballots(U, H)$) and there is an uncounted ballot for some candidate C ($uncounted-ballot(C)$) and C ’s vote count is currently N ($hopeful(C, N)$), then C ’s vote count is updated to $N+1$ ($hopeful(C, N+1)$) and we continue counting ballots ($count-ballots(U-1, H)$).’

count/done: ‘If we are in the process of counting ballots and no uncounted ballots remain ($count-ballots(0, H)$), then we have finished counting ballots and should now begin determining which candidate has the greatest vote total ($determine-max(H)$).’

There are several key observations to be made.

Use of Linearity. The linear resource discipline is crucial for the `count/run` and `count/done` axioms to adequately specify SW-FPTP ballot counting.

First, the *hopeful* propositions that record candidates' vote counts are treated as linear resources. That is, they are not prefixed with the unrestricted modality, $!$, which would escape the resource discipline. Being linear resources allows the vote counts to be mutable. This is fundamental to the correctness of the `count/run` axiom: whenever a ballot for candidate C is counted, the record of C 's vote count ($\text{hopeful}(C, N)$) is consumed and is replaced with a new, updated record ($\text{hopeful}(C, N+1)$). If these *hopeful* records were not linear resources, then they would not be consumed and the old vote counts would persist alongside the new count, causing untold confusion.

Second, and equally crucial, *uncounted-ballot*(C) is treated as a linear resource. Consequently, ballots are consumed whenever they are counted by the `count/run` axiom. If ballots were not linear, they would not be consumed upon being counted; they would effectively remain in the ballot box, leaving open the possibility that a ballot could be counted more than once. (Note that we need not preserve the ballot in a counted form, e.g., as in *counted-ballot*(C); the candidate's vote count is sufficient to reconstruct the ballots that were cast.)

Treating uncounted ballots as linear resources also provides a further benefit. Because linear logic demands that resources be used *at least* once, the specification framework itself ensures that all ballots are eventually counted by `count/run`. This is a strong guarantee that is provided to the specification for free!

Tracking the Number of Uncounted Ballots. The reader may wonder why we bother to maintain the invariant that there are U uncounted ballots and H hopeful candidates remaining whenever *count-ballots*(U, H) holds. For example, couldn't one just use

$$\begin{aligned} & \text{count-ballots}' \otimes \\ & \text{uncounted-ballot}(C) \otimes \text{hopeful}(C, N) \\ & \multimap \{ \text{hopeful}(C, N+1) \otimes \text{count-ballots}' \} \end{aligned}$$

in place of the `count/run` axiom?

The answer is that this invariant makes it possible to decide when there are no more ballots to count: there are no more ballots exactly when *count-ballots*($0, H$) holds. This forms the basis for `count/done`. Without tracking this information, `count/done` would need to rely on extra-logical machinery, such as negation-as-failure, to check for the absence of *uncounted-ballot*(C). And, extra-logical machinery would forfeit the benefits of a purely logical specification as discussed in Sect. [II](#).

The *count-ballots* and *determine-max* predicates track the number of remaining hopefuls for similar reasons. Doing so provides a purely logical way to determine when exactly one hopeful remains, forming the basis for `max/done`, as described below.

3.2 Determining the Winner with Max/Run and Max/Done

The max/run and max/done axioms characterize a random tournament for determining which candidate has the greatest vote total. An intuitive reading of the axioms is as follows:

max/run: ‘If we are in the process of determining who has the greatest vote total ($\text{determine-max}(H)$) and there are at least two candidates C and C' that remain hopeful, with vote totals N and N' respectively, ($\text{hopeful}(C, N) \otimes \text{hopeful}(C', N')$) and C 's vote total is larger ($!(N \geq N')$), then C remains a hopeful ($\text{hopeful}(C, N)$) and C' is defeated ($!\text{defeated}(C')$) and we continue to determine who has the greatest vote total ($\text{determine-max}(H-1)$).’

max/done: ‘If we are in the process of determining who has the greatest vote total and only one candidate remains a hopeful ($\text{determine-max}(1)$) and that candidate is C ($\text{hopeful}(C, N)$), then C is declared the winner ($!\text{elected}(C)$).’

Two key points benefit from further explanation.

Use of the Unrestricted Modality, !. The unrestricted modality is used strategically at three points in the max/run and max/done axioms.

First, the max/run axiom includes $!(N \geq N')$ to ensure that C 's vote total is no smaller than that of C' . As presented in Sect. 2.1, the unrestricted modality, $!$, denotes a fact that remains true regardless of how the system's state evolves. Because natural number inequalities are true independent of the voting system's state, the use of $!$ here is justified.

Second and third, the max/run and max/done axioms use $!\text{defeated}(C')$ and $!\text{elected}(C)$, respectively, to reflect changes in a candidate's status. Just as $!$ denotes a fact that remains true regardless of how the system's state evolves, these uses of $!$ express that once a candidate is either defeated or elected her status (in that election) never changes.

No Axiom Dual to Max/Run. At first glance, it may be surprising that there is no axiom, dual to max/run , for the case $!(N \leq N')$. In fact, max/run itself handles this case. For a fixed pair of *hopeful* assumptions, the premise $\text{hopeful}(C, N) \otimes \text{hopeful}(C', N')$ can be instantiated in two ways: one for each permutation of those two assumptions. At least one of these will satisfy the inequality $!(N \geq N')$, and so the single max/run axiom suffices.

4 Viewing Specifications as Linear Logic Programs

Thus far we have given a static specification of SW-FPTP as a collection of linear logical axioms. However, viewed through the lens of linear logic programming, such specifications can also be seen as rules defining a forward-chaining,

committed-choice linear logic program [13,14]. Thus, specifications can be directly executed using a logic programming engine. (As a demonstration, a transliteration of the SW-FPTP specification into Celf source code is available at <http://www.itu.dk/~carsten/files/voteid2011.tgz>.)

Rules of such logic programs are essentially multiset rewriting rules [3]. For instance, max/run can be seen as the following multiset rewriting rule: choose any two *hopeful* terms and replace the one having a smaller vote count with a corresponding *!defeated* term. Given similar interpretations of the other axioms, the SW-FPTP algorithm can be run by issuing logic programming queries.

Example 1. Consider the following election scenario. Three candidates, a, b, and c, are running for office. We model this with three linear resources that initialize the candidates' vote counts to 0: $\text{hopeful}(a, 0)$, $\text{hopeful}(b, 0)$, and $\text{hopeful}(c, 0)$. Each uncounted ballot is also modeled as a linear resource:

$$\begin{aligned} &\text{uncounted-ballot}(a), \text{uncounted-ballot}(b), \text{uncounted-ballot}(a), \\ &\text{uncounted-ballot}(c), \text{uncounted-ballot}(b), \text{uncounted-ballot}(a). \end{aligned}$$

To initiate the execution, we add the resource $\text{count-ballots}(6, 3)$, which is indexed by the number of uncounted ballots and the number of *hopefuls*. The execution consists of two phases: in the first phase, the votes are tallied, and in the second phase, the candidate who has the most votes is determined. At each step of the execution, one of the SW-FPTP axioms is applied as a multiset rewriting rule, following the rules of linear logic. First, count/run fires exactly six times, once for each ballot. Next, the rule count/done fires, and the execution commences with the second phase. Determining the winner requires two comparisons, which means that max/run fires twice. And finally, the max/done rule fires, announcing a as the winner and concluding the execution.

The logic programming approach therefore provides two key benefits. First, and most importantly, there is no need to verify the executable code against a separate formal specification: the code and specification are one and the same! (Only a linear logic proof engine needs to be trusted as an interpreter.) This benefit cannot be overemphasized, and is a direct result of choosing linear logic as the high-level, yet fully rigorous, specification language.

Second, the traces of rewriting steps that are produced by the logic programming engine provide an immediate means for auditing the election. Because the traces are, in fact, proof objects in linear logic, auditing is easy: a lightweight linear logic proof checker can formally verify the validity of a trace. In particular, costly recounts become unnecessary because the verifiable traces record each step of vote counting.

5 Single Transferable Vote in Linear Logic

To show that linear logic is robust enough to be used for specifying complex voting systems, we now turn our attention to single transferable vote (STV).

In STV, each voter casts a ballot that lists candidates in order of the voter’s preference. To be elected, a candidate must reach a threshold, or quota, of votes. For the purposes of this paper, the particular choice of quota is arbitrary. Because it is commonly used in practice, we choose the Droop quota,

$$\text{quota} = \frac{\#\text{ballots}}{\#\text{seats} + 1} + 1 ,$$

however any quota could easily be substituted. Once the quota is computed, the ballots are counted and the following rules are repeated until all open seats are filled.

1. If a candidate has enough votes to meet the quota, she is declared elected. Any surplus votes for this candidate are transferred.
2. If all ballots have been assigned to candidates and no candidate meets the quota, then the candidate with the fewest votes is eliminated and her votes are transferred. If several candidates tie for the fewest votes, one is eliminated at random.
3. When a vote is transferred, it is assigned to the hopeful candidate with the next highest preference listed on that ballot. That is, candidates that are already elected or defeated do not receive transferred votes.
4. If, at any point, there are at least as many open seats as hopeful candidates remaining, then all remaining hopefuls become elected.

5.1 A Linear Logical Specification of Single Transferable Vote

For our specification of STV, we must introduce several predicates, which are summarized in Table 2. The *uncounted-ballot*, *counted-ballot*, *hopeful*, *defeated*, *elected*, *quota*, and *winners* predicates characterize the ballot box, candidates’ statuses, and the election’s state. The *elect-all*, *defeat-min*, *defeat-min'*, *transfer*, and *begin* predicates are used to indicate progress through the STV algorithm’s phases. Finally, *minimum* is an auxiliary predicate used in determining a candidate with the fewest votes. (We again assume the usual ordering predicates on natural numbers, such as $!(N \geq N')$.)

The linear logical axioms that specify STV are given in Fig. 2. Several of these axioms pattern-match on the shape of a list of candidates. Following standard convention, we use $[\]$ to stand for the empty list and $[C \mid L]$ to stand for the non-empty list with head C and tail L . (We again follow the convention that universal quantification is implicit for variables written in upper case.)

These axioms faithfully encode STV in a concise and elegant fashion—rather than requiring hundreds or thousands of lines of imperative source code, our full STV specification fits on a single page! To make plain the close correspondence of the axioms with the natural language description of STV used in current practice, we will now walk through their meanings.

Beginning the STV Algorithm. The `begin/1` axiom describes the initial step of the STV algorithm: the Droop quota is computed and recorded. Ballot counting is initiated, with no candidates having been declared winners.

Table 2. Descriptions of predicates used in the STV specification

Predicate	Meaning
<i>uncounted-ballot</i> (C, L)	An uncounted ballot with highest preference for candidate C and list L of lower preferences.
<i>counted-ballot</i> (C, L)	A ballot counted for candidate C , with list L of lower preferences.
<i>hopeful</i> (C, N)	Candidate C is not yet defeated nor elected, and N ballots have been counted for C thus far.
<i>!defeated</i> (C)	Candidate C has been (and will remain) defeated.
<i>!elected</i> (C)	Candidate C has been (and will remain) elected.
<i>!quota</i> (Q)	Q votes are needed to be elected.
<i>winner</i> s(W)	The candidates in list W have been elected thus far.
<i>begin</i> (S, H, U)	Token to signal that the STV algorithm should begin running. There are S seats up for election, H hopeful candidates, and U ballots cast.
<i>count-ballots</i> (S, H, U)	Token to indicate that the algorithm is counting ballots, and that there are S open seats, H hopeful candidates, and U uncounted ballots remaining.
<i>!elect-all</i>	Token to indicate that there are more open seats than hopefuls remaining; all remaining hopefuls should become elected.
<i>defeat-min</i> (S, H, M)	Token to indicate that the algorithm is in the first step of determining a candidate who has the fewest votes. There are S open seats, H hopeful candidates, and M potential minimums remaining.
<i>defeat-min'</i> (S, H, M)	Token to indicate that the algorithm is in the second step of determining a candidate who has the fewest votes. There are S open seats, H hopeful candidates, and M potential minimums remaining.
<i>minimum</i> (C, N)	Candidate C 's vote count of N is a potential minimum.
<i>transfer</i> (C, N, S, H, U)	Token to indicate that newly defeated candidate C 's remaining N votes are being transferred. There are S open seats, H hopeful candidates, and U uncounted ballots.

begin/1 :
begin(S, H, U) \otimes
 $!(Q = U/(S + 1) + 1)$
 $\rightarrow \{!quota(Q) \otimes winners([\])$
 $\quad count-ballots(S, H, U)\}$

count/1 :
count-ballots(S, H, U) \otimes
uncounted-ballot(C, L) \otimes
hopeful(C, N) \otimes
 $!quota(Q) \otimes !(N+1 < Q)$
 $\rightarrow \{counted-ballot(C, L) \otimes$
 $\quad hopeful(C, N+1) \otimes$
 $\quad count-ballots(S, H, U-1)\}$

count/2 :
count-ballots(S, H, U) \otimes
uncounted-ballot(C, L) \otimes
hopeful(C, N) \otimes
 $!quota(Q) \otimes !(N+1 \geq Q) \otimes$
 $!(S \geq 1) \otimes winners(W)$
 $\rightarrow \{counted-ballot(C, L) \otimes$
 $\quad !elected(C) \otimes winners([C | W]) \otimes$
 $\quad count-ballots(S-1, H-1, U-1)\}$

count/3.1 :
count-ballots(S, H, U) \otimes
uncounted-ballot($C, [C' | L]$) \otimes
elected(C)
 $\rightarrow \{uncounted-ballot(C', L) \otimes$
 $\quad count-ballots(S, H, U)\}$

count/3.2 :
count-ballots(S, H, U) \otimes
uncounted-ballot($C, [C' | L]$) \otimes
defeated(C)
 $\rightarrow \{uncounted-ballot(C', L) \otimes$
 $\quad count-ballots(S, H, U)\}$

count/4.1 :
count-ballots(S, H, U) \otimes
uncounted-ballot($C, [\]$) \otimes
elected(C)
 $\rightarrow \{count-ballots(S, H, U-1)\}$

count/4.2 :
count-ballots(S, H, U) \otimes
uncounted-ballot($C, [\]$) \otimes
defeated(C)
 $\rightarrow \{count-ballots(S, H, U-1)\}$

count/5 :
count-ballots($S, H, 0$) $\otimes !(S < H)$
 $\rightarrow \{defeat-min(S, H, 0)\}$

count/6 :
count-ballots($S, H, 0$) $\otimes !(S \geq H)$
 $\rightarrow \{elect-all\}$

defeat-min/1 :
defeat-min(S, H, M) \otimes
hopeful(C, N)
 $\rightarrow \{minimum(C, N) \otimes$
 $\quad defeat-min(S, H-1, M+1)\}$

defeat-min/2 :
defeat-min($S, 0, M$)
 $\rightarrow \{defeat-min'(S, 0, M)\}$

defeat-min'/1 :
defeat-min'(S, H, M) \otimes
minimum(C, N) \otimes
minimum(C', N') \otimes
 $!(N \leq N')$
 $\rightarrow \{minimum(C, N) \otimes$
 $\quad hopeful(C', N') \otimes$
 $\quad defeat-min'(S, H+1, M-1)\}$

defeat-min'/2 :
defeat-min'($S, H, 1$) \otimes
minimum(C, N)
 $\rightarrow \{!defeated(C) \otimes$
 $\quad transfer(C, N, S, H, 0)\}$

transfer/1 :
transfer(C, N, S, H, U) \otimes
counted-ballot(C, L)
 $\rightarrow \{uncounted-ballot(C, L) \otimes$
 $\quad transfer(C, N-1, S, H, U+1)\}$

transfer/2 :
transfer($C, 0, S, H, U$)
 $\rightarrow \{count-ballots(S, H, U)\}$

elect-all/1 :
elect-all \otimes
hopeful(C, N) $\otimes winners(W)$
 $\rightarrow \{!elected(C) \otimes winners([C | W])\}$

cleanup/1 :
elect-all \otimes
counted-ballot(C, L)
 $\rightarrow \{1\}$

Fig. 2. A linear logical specification of single transferable vote

Counting the Ballots

- `count/1` describes counting a ballot that does not cause its candidate, C , to reach the quota: C 's vote total increases and ballot counting continues.
- `count/2` describes counting a ballot that causes its candidate, C , to finally reach the quota. C becomes elected, being a hopeful no longer, and is added to the list of winners. Any ballots remaining uncounted for C constitute C 's vote surplus; the surplus is randomly selected because ballots are counted in a random order. After C is elected, ballots continue to be counted.
- `count/3.1`, `count/3.2`, `count/4.1`, and `count/4.2` express that no more ballots are counted for candidates that are already either elected or defeated. The ballots transfer to the next highest preference; if none exists, the ballot is consumed—that is, the vote is wasted.
- Finally, `count/5` and `count/6` describe what happens when there are no more ballots to count. If there are fewer open seats than hopefuls remaining (`count/5`), then a candidate with the fewest votes is defeated; the generated *defeat-min* token begins this process. Otherwise, if there are at least as many open seats as hopefuls (`count/6`), then all remaining hopefuls are elected.

Defeating a Candidate with the Fewest Votes

- `defeat-min/1` labels all hopeful candidates as potential minimums. When there are no more hopefuls to label (i.e., when the H counter reaches 0), `defeat-min/2` transitions to the second phase of defeating a candidate.
- `defeat-min'/1` and `defeat-min'/2` describe a random tournament for finding, among the potential minimums, a candidate with the fewest votes. Candidates not selected as the minimum are restored to their hopeful status (`defeat-min'/1`). When only one candidate is a potential minimum (i.e., when the M counter reaches 1), that candidate must have the fewest votes; she is defeated and the process of transferring her votes begins (`defeat-min'/2`).

Transferring a Defeated Candidate's Votes

- `transfer/1` expresses that ballots counted for a newly defeated candidate, C , are returned to the ballot box as uncounted ballots. As `transfer/2` shows, when the N counter reaches 0, these ballots will be re-counted. Because C is now defeated, re-counting these ballots will effectively transfer them to the next highest preference, if one exists (`count/3.2` and `count/4.2`).

Finishing the STV Election

- `elect-all/1` expresses that the STV algorithm finishes by electing all remaining hopefuls. (Note that there may possibly be no remaining hopefuls at this point.) Because this is the last step of the STV algorithm, we may think of this step as continuing forever, idling once all remaining hopefuls have been elected. This justifies the use of the ! modality here and also in `count/6`.

- When the STV algorithm finishes, the counted ballots will remain as linear resources. The resource discipline of linear logic demands that these be used once. Therefore, the `cleanup/1` axiom consumes any remaining ballots. This is safe because the STV algorithm has already filled all seats.

5.2 Viewing the STV Specification as a Linear Logic Program

As we did for SW-FPTP (Sect. 4), we can view the STV specification as a linear logic program. This provides executable code for STV with the same benefits as for SW-FPTP before: a close correspondence between code and specification, with no separate verification needed, and verifiable traces to audit the election. As a demonstration, a transliteration of the STV specification into Celf source code is available at <http://www.itu.dk/~carsten/files/voteid2011.tgz>.

For STV, because of the coercion problem [2], only trusted individuals should be given access to these traces. Determining the right way to adapt the cryptographic solutions of the coercion problem to the logic programming approach is an intriguing area for future work.

6 Proving the Specifications Correct

From the preceding discussions of their axioms, it should be clear that the specifications correspond to the SW-FPTP and STV protocols, respectively. However, we would like to rigorously prove these claims of correctness. Following ideas for other logical specifications [10,9], we can prove such properties by straightforward induction on the specification’s operational semantics. To demonstrate this technique, we will sketch correctness proofs for our protocol specifications.

6.1 Correctness of the SW-FPTP Specification

To prove that the SW-FPTP specification is correct, we must show that all executions elect a candidate who has at least as many votes as all other candidates. First, we establish invariants that characterize valid states in executions of the SW-FPTP protocol.

Lemma 1. *All SW-FPTP axioms preserve the following state invariants.*

- *Exactly one of the following holds:*
 - *there is exactly one assumption $\text{count-ballots}(U, H)$,*
 - *there is exactly one assumption $\text{determine-max}(H)$, or*
 - *there are no assumptions $\text{count-ballots}(U, H)$ and $\text{determine-max}(H)$.*
- *For each candidate C , exactly one of the following holds:*
 - *$\text{!elected}(C)$ and there are no $\text{uncounted-ballot}(C)$ assumptions,*
 - *$\text{!defeated}(C)$ and there are no $\text{uncounted-ballot}(C)$ assumptions, or*
 - *there is exactly one assumption $\text{hopeful}(C, N)$.*

- If $\text{count-ballots}(U, H)$, then
 - there are no $\text{!elected}(C)$ or $\text{!defeated}(C)$ assumptions,
 - there are H assumptions of the form $\text{hopeful}(C, N)$, and
 - there are U assumptions of the form $\text{uncounted-ballot}(C)$.
- If $\text{determine-max}(H)$, then
 - there are no $\text{!elected}(C)$ assumptions,
 - there are H assumptions of the form $\text{hopeful}(C, N)$, and
 - there are no assumptions of the form $\text{uncounted-ballot}(C)$.
- If neither $\text{count-ballots}(U, H)$ nor $\text{determine-max}(H)$, then
 - either there is exactly one candidate C for which $\text{!elected}(C)$, or there are no $\text{!elected}(C)$ or $\text{!defeated}(C)$ assumptions; and
 - there are no $\text{hopeful}(C, N)$ assumptions.

Proof. Directly by case analysis on the SW-FPTP axioms.

Next, let a hopeful candidate C 's vote total be the sum of the number of $\text{uncounted-ballot}(C)$ assumptions and the unique value N for which $\text{hopeful}(C, N)$ holds. Correctness of our SW-FPTP specification then follows:

Theorem 1. *For all partial executions, the following hold for each candidate C :*

- If $\text{hopeful}(C, N_1)$ holds at the end of the partial execution, then, at the beginning of the partial execution, $\text{hopeful}(C, N_0)$ holds and C 's vote total is the same as it will be at the end of the partial execution.
- If $\text{!defeated}(C)$ holds at the end of the partial execution, then, at the beginning of the partial execution, either:
 1. $\text{!defeated}(C)$ holds, or
 2. $\text{hopeful}(C, N_0)$ holds and C 's vote total is no larger than the vote total of some candidate C' for which $\text{hopeful}(C', N'_0)$ holds.
- If $\text{!elected}(C)$ holds at the end of the partial execution, then, at the beginning of the partial execution, either:
 1. $\text{!elected}(C)$ holds, or
 2. $\text{hopeful}(C, N_0)$ holds and C 's vote total is at least as large as the vote totals of all candidates C' for which $\text{hopeful}(C', N'_0)$ holds.

Proof. By induction on the specification's operational semantics (that is, the length of the execution's trace), using the invariants from Lemma [□](#).

6.2 Correctness of the STV Specification

Unfortunately, unlike for SW-FPTP, there is no independent, formal model of STV against which we might prove the full correctness of our specification. This, in a sense, is a primary benefit of our linear logical approach: the specification's operational semantics serves as a formal model of STV.

Despite the absence of an independent model, we *can* prove properties that we expect the putative STV specification to possess. For instance, when the STV protocol finishes, it should be that the number of elected candidates exactly equals the number of seats, **SEATS**, that were to be filled by the election (assuming that, initially, there are at least as many candidates as seats). To prove this, we follow a similar strategy as for SW-FPTP.

Let $\#elected$ and $\#hopeful$ be the number of distinct candidates C for which $!elected(C)$ and $hopeful(C, N)$, for some N , hold, respectively. We first establish invariants that characterize valid states in executions of the STV protocol.

Lemma 2. *All STV axioms preserve the following state invariants provided that $!begin(S, H, U)$ implies $S = SEATS$, $\#elected = 0$, $H = \#hopeful$, and $S \leq H$.*

- If $count\text{-}ballots(S, H, U)$, then $S = SEATS - \#elected$, $H = \#hopeful$, and $S \leq H$.
- If $defeat\text{-}min(S, H, M)$ or $defeat\text{-}min'(S, H, M)$, then $S = SEATS - \#elected$, $H = \#hopeful$, and $S < H + M$.
- If $transfer(C, N, S, H, U)$, then $S = SEATS - \#elected$, $H = \#hopeful$, and $S \leq H$.
- If $!elect\text{-}all$, then $\#elected + \#hopeful = SEATS$.

Proof. Directly by case analysis on the STV axioms.

Then, because $!elect\text{-}all$ holds and no hopefuls remain when the execution concludes, the property is immediate:

Corollary 1. *For all executions of the STV specification, the final state satisfies $\#elected = SEATS$.*

Because STV is the more complicated protocol, it may seem counterintuitive that the STV invariants stated in Lemma 2 are simpler than the SW-FPTP invariants of Lemma 1. In fact, the invariants of Lemma 2 are not the strongest invariants that we could prove, but suffice to prove the property of interest.

7 Conclusion

In this paper, we have promoted linear logic as a practical, mathematical language for the rigorous specification and implementation of voting protocols. We have demonstrated this methodology on the SW-FPTP and STV protocols. Linear logic yields concise specifications, implementations for free, auditable executions, and, perhaps most importantly, formal proofs of operational correctness.

In future work, we plan to extend our linear logic and its logic programming engine with modalities for knowledge, possession, and secrecy, which will greatly increase the expressive strength of the logic. For example, an epistemic modality would give logical meaning to the secrecy properties of encryption. Modal logics of Garg et al. [10,11] will serve as a foundation on which to build. We also intend to develop techniques for reasoning about the security of voting protocols specified in linear logic, such as for proving privacy and coercion-resistance.

Acknowledgements. This material is based upon work supported by a National Science Foundation Graduate Research Fellowship for the first author. The second author was supported in part by grant 10-092309 from the Danish Council for Strategic Research, Programme Commission on Strategic Growth Technologies, and by grant 10-093084 from the Danish Council for Independent Research, Natural Sciences. The authors also wish to thank the anonymous reviewers for their invaluable comments.

References

1. Andreoli, J.M.: Logic programming with focusing proofs in linear logic. *Journal of Logic and Computation* 2(3), 297–347 (1992)
2. Benaloh, J., Moran, T., Naish, L., Ramchen, K., Teague, V.: Shuffle-sum: Coercion-resistant verifiable tallying for STV voting. *IEEE Transactions on Information Forensics and Security* 4(4), 685–698 (2009)
3. Cervesato, I., Scedrov, A.: Relating state-based and process-based concurrency through linear logic. *Information & Computation* 207(10), 1044–1077 (2009)
4. Chang, B.Y.E., Chaudhuri, K., Pfenning, F.: A judgmental analysis of linear logic. *Tech. Rep. CMU-CS-03-131R*, Carnegie Mellon University (December 2003)
5. Chaum, D., Ryan, P.Y.A., Schneider, S.: A Practical Voter-Verifiable Election Scheme. In: De Capitani di Vimercati, S., Syverson, P.F., Gollmann, D. (eds.) *ESORICS 2005*. LNCS, vol. 3679, pp. 118–139. Springer, Heidelberg (2005)
6. Clocksin, W.F., Mellish, C.S.: *Programming in Prolog*, 5th edn. Springer (2003)
7. Cochran, D., Kiniry, J.: Vótáil: A formally specified and verified ballot counting system for Irish PR-STV elections. In: Beckert, B., Marché, C. (eds.) *Pre-proceedings of the International Conference on Formal Verification of Object-Oriented Software*, Paris, France (June 2010)
8. Delaune, S., Kremer, S., Ryan, M.: Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security* 17(4), 435–487 (2009)
9. DeYoung, H., Pfenning, F.: Reasoning about the consequences of authorization policies in a linear epistemic logic. In: Cortier, V., Shmatikov, V. (eds.) *Proceedings of the Workshop on Foundations of Computer Security*, Los Angeles, California (August 2009)
10. Garg, D., Bauer, L., Bowers, K.D., Pfenning, F., Reiter, M.K.: A Linear Logic of Authorization and Knowledge. In: Gollmann, D., Meier, J., Sabelfeld, A. (eds.) *ESORICS 2006*. LNCS, vol. 4189, pp. 297–312. Springer, Heidelberg (2006)
11. Garg, D., Pfenning, F.: A proof-carrying file system. In: *31st IEEE Symposium on Security and Privacy*, pp. 349–364. IEEE Computer Society Press, Oakland (2010)
12. Girard, J.Y.: Linear logic. *Theoretical Computer Science* 50(1), 1–102 (1987)
13. López, P., Pfenning, F., Polakow, J., Watkins, K.: Monadic concurrent linear logic programming. In: Barahona, P., Felty, A.P. (eds.) *Proceedings of the 7th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming*, pp. 35–46. ACM Press, Lisbon (2005)
14. Schack-Nielsen, A., Schürmann, C.: Celf – A Logical Framework for Deductive and Concurrent Systems (System Description). In: Armando, A., Baumgartner, P., Dowek, G. (eds.) *IJCAR 2008*. LNCS (LNAI), vol. 5195, pp. 320–326. Springer, Heidelberg (2008)
15. Wadler, P.: A Taste of Linear Logic. In: Borzyszkowski, A.M., Sokolowski, S. (eds.) *MFCS 1993*. LNCS, vol. 711, pp. 185–210. Springer, Heidelberg (1993)
16. Watkins, K., Cervesato, I., Pfenning, F., Walker, D.: A concurrent logical framework I: Judgments and properties. *Tech. Rep. CMU-CS-02-101*, Carnegie Mellon University (2002) (revised May 2003)

Efficient Vote Authorization in Coercion-Resistant Internet Voting

Michael Schlapfer¹, Rolf Haenni², Reto Koenig^{2,3}, and Oliver Spycher^{2,3}

¹ ETH Zurich, CH-8092 Zurich, Switzerland
michschl@inf.ethz.ch

² Bern University of Applied Sciences, CH-2501 Biel, Switzerland
{rolf.haenni, reto.koenig, oliver.spycher}@bfh.ch

³ University of Fribourg, CH-1700 Fribourg, Switzerland
{reto.koenig, oliver.spycher}@unifr.ch

Abstract. Some years ago, Juels et al. introduced the first coercion-resistant Internet voting protocol. Its basic concept is still the most viable approach to address voter coercion and vote selling in Internet voting. However, one of the main open issues is its unrealistic computational requirements of the quadratic-time tallying procedure. In this paper, we examine the cause of this issue, namely the authorization of votes, and summarize the most recent proposals to perform this step in linear time. We explain the key underlying concepts of these proposals and introduce a new protocol based on anonymity sets. The size of these anonymity sets serves as an adjustable security parameter, which determines the degree of coercion-resistance. The main advantage of the new protocol is to move computational complexity introduced in recent works from the voter side to the tallying authority side.

1 Introduction

Remote voting over the Internet gains increasing attention as many governments aim at providing their citizens with electronic voting services. Although tremendous effort was put in research to understand the various aspects involved in electronic voting, no widely accepted solution to overcome all the security problems has been presented so far. One particular problem, namely voter coercion, was introduced in 2005 by Juels et al. in [13]. They propose a coercion-resistant Internet voting protocol to which we will simply refer as JCJ. Their protocol has been widely discussed and examined in the literature, and its basic concept still seems to be the most viable solution to address the voter coercion and vote selling problems. As appealing the approach may look in theory, it leaves some critical issues unanswered, for example the board-flooding problem or the quadratic running time of the tallying procedure. As we will not address the board-flooding problem in this paper, we refer to some of the most recent proposals in the literature [14].

This paper deals with the latter problem, i.e., the quadratic running time of the tallying procedure. We particularly focus on the two main building blocks of JCJ-based protocols, namely the elimination of *duplicate votes* and the detection of *fake votes*. These components are responsible for the expensive computations during tallying. Together with the elimination of invalid votes (those with invalid zero-knowledge proofs),

we generally refer to these steps as *vote authorization*. We propose a modified protocol based on anonymity sets to address the efficiency problems of these building blocks and compare our protocol to some recent proposals for efficient vote authorization. In comparison with the closest recent protocol [6, 7], our approach is more expensive for the tallying authorities, but less expensive for voters. We consider this as an important property, because the computational resources on the voters’ side are usually limited (e.g. in Internet voting, low-power devices or slow interpreted languages might be used by voters).

In Section 2, we first introduce the JCJ protocol, its critical building blocks, and recent improvements. In Section 3, we propose a modified protocol for efficient vote authorization. We compare the performance of our approach to existing protocols in Section 4. Finally, we summarize our conclusions and suggest some future work in Section 5.

2 Coercion-Resistant Internet Voting

Coercion-resistant Internet voting protocols follow in general the phases depicted in Figure 1. Prior to any voting event, eligible voters must register with the authority to get their credentials for participating at future elections. This phase is individual for the voters and usually carried out only once for a number of subsequent voting events. For every election, the following phases are repeatedly performed: *election setup*, *vote casting*, *vote authorization*, *tallying*. Note that some protocols allow to partly carry out vote authorization already during the vote casting phase.

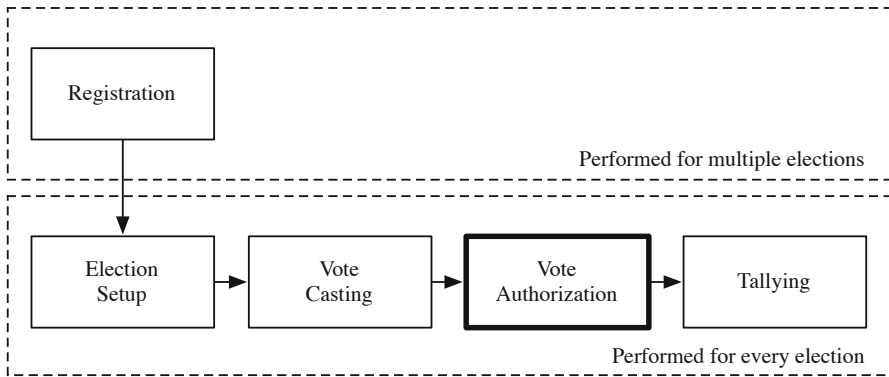


Fig. 1. Phases of coercion-resistant Internet voting protocols

While efficient and scalable solutions exist for most of the above phases, the vote authorization phase—as proposed by JCJ—requires computing capacities which grow quadratically in the number of votes cast. In a large-scale election setting such as nationwide parliamentary elections, this leads to unrealistic performance requirements. In the following, we explain the vote authorization phase in more detail, and then we provide a summary of the original JCJ protocol and of some recently proposed improvements.

2.1 Vote Authorization

Voting from a remote private place is inherently problematic with respect to privacy, because no privacy-preserving voting booth is used to protect it. Therefore, other measures must be taken to protect voters from coercion and to prevent vote selling. The main measures for this in JCJ are (a) the ability of coerced voters to create and cast fake votes, the coercer cannot distinguish from valid votes, and (b) the ability for voters to cast multiple votes, of which only one will be counted. Hence, various types of unauthorized votes may appear in the electronic ballot box and must thus be excluded before tallying. In some protocol descriptions, this phase is called *pre-tallying phase*, whereas sometimes it is implicitly included in the tallying phase. We call it *vote authorization phase*, which consists of the following three consecutive vote elimination steps:

Invalid Votes Elimination. The ballots created and cast by the voters usually include cryptographic zero-knowledge proofs for various purposes, for example for the correct construction of the encryptions, the correctness of the selected candidate choices, or the knowledge of the plaintexts. In the first step of the vote authorization phase, the authorities verify the validity of the proofs to exclude ballots with invalid proofs from further processing.

Duplicate Votes Elimination. In the second step, among all ballots that were cast with the same credential, exactly one is chosen for further processing according to some policy. In the “last-vote-counts” policy, for example, the last valid vote cast is selected to be included in the tallying, whereas all other votes from the same credential are excluded from further processing. This enforces the “one-voter-one-vote” principle.

Fake Votes Elimination. To conclude the vote authorization phase, all remaining fake votes have to be detected and removed. For this, the authorities need to check whether or not the ballots include credentials from eligible voters. Only the ballots passing this test are kept for the final tallying. Note that before eliminating fake votes, they must be unlinked from the actual votes cast. Otherwise, a coercer or vote buyer could easily detect the voter’s attempt of not fulfilling the demands. The unlinking is usually achieved by shuffling the votes in a verifiable (re-encryption) mix-net.

2.2 The JCJ Protocol

In the following paragraphs, we briefly explain the phases of the JCJ protocol. Since we focus on the main building blocks, we settle for a semi-formal style of exposition. In particular, we do not thoroughly explain well-known cryptographic techniques. Furthermore, we assume the application of publicly verifiable group threshold mechanisms whenever registering or tallying authorities perform joint computations, even if the text might suggest a single entity. All ciphertexts are ElGamal encryptions over a pre-established multiplicative cyclic group $(\mathcal{G}_q, \cdot, 1)$ of order q , for which the decisional Diffie-Hellman problem (DDHP) is assumed to be hard. Note that the authors of the JCJ

protocol propose a modified version of ElGamal encryption for their formal proofs to work. The discussion in this paper is based on a simplified version with ordinary ElGamal encryptions as it is used by CIVITAS [9].

Registration. The *registrars* establish the random credential $\sigma \in \mathcal{G}_q$ and pass it to the voter via an *untappable channel*. Additionally, they append a public credential, i.e., a randomized encryption $S = \text{Enc}_\varepsilon(\sigma, \alpha_S)$ of σ , to the voter's entry in the public voter roll, which is part of a public bulletin board. Value α_S denotes the encryption's randomness, and ε stands for the tallying authorities' common public key. Assuming a majority of trustworthy registrars, in the end only the voter will know σ and no one will know α_S .

Vote Casting. The voter identifies a choice c from the available set of valid choices (or candidates) \mathcal{C} . To cast the vote, the encryptions $A = \text{Enc}_\varepsilon(\sigma, \alpha_A)$ and $B = \text{Enc}_\varepsilon(c, \alpha_B)$ are posted to the public bulletin board, via an anonymous channel. The pair (A, B) must be accompanied by two non-interactive zero-knowledge proofs (NIZKP), one to prove knowledge of σ and one to prove $c \in \mathcal{C}$. Requiring the first proof prevents attackers from casting unauthorized votes by re-encrypting entries from the voter roll (recall that α_S is not known to anyone). Since each authorized vote on the voting board will be decrypted during the tallying phase, the second proof is needed to prevent coercers from forcing voters to select $c \notin \mathcal{C}$ according to some prescribed pattern, thus obtaining a receipt as described in [10].

To circumvent coercion, the voter can deceive the coercer by posting a fake vote to the voting board. To do so, the voter simply claims some arbitrary $\sigma' \in \mathcal{G}_q$ to be the proper credential and uses it to compute A . The encrypted vote B is computed according to the coercer's preference and the plaintexts of A and B are revealed to justify compliance. Alternatively, the voter can even let the coercer compute A and B and cast the vote using σ' .

Vote Authorization. At the end of the vote casting phase, the voting board contains a certain number N of votes cast, of which not all might make it to the final tally. First, the authorities verify all NIZKPs that were cast along with the votes. If a proof does not hold for a vote (A, B) , it is marked accordingly and excluded from further processing (*invalid votes elimination*). Then the tallying authorities need to filter out votes that were cast multiple times with a proper credential (*duplicate votes elimination*) and votes that were cast with a fake credential (*fake votes elimination*). For both tasks, the authors of JCY propose the application of *plaintext equivalence tests* (PET) [11]. Given two ElGamal encryptions $X = \text{Enc}_\varepsilon(x, \alpha_X)$ and $Y = \text{Enc}_\varepsilon(y, \alpha_Y)$, the group threshold algorithm $\text{PET}(X, Y)$ returns *true* for $x = y$ and *false* for $x \neq y$, without revealing any information on x or y [1].

Tallying. At the end, i.e., after eliminating all unauthorized votes, all remaining votes are jointly decrypted and counted. The final result is published.

¹ A common way of performing PET in a homomorphic encryption scheme is to check whether the decryption of $(X/Y)^z$ equals 1 for some random value z .

2.3 Improvements of the JCJ Protocol

Several proposals for improving the quadratic running-time of the tallying procedure in JCJ exist in the recent literature. In this subsection, we give a short overview of these developments. Two of them will be described in more details, as some of their ideas will re-appear in the description of our contribution in Section 3.

Smith, Weber [17, 19, 20]. Instead of applying $\text{PET}(A_i, A_j)$, $1 \leq i, j \leq N$, on all pairs of distinct ciphertexts for removing duplicates, both Smith and Weber suggested computing and decrypting $A_1^z = \text{Enc}_\varepsilon(\sigma_1^z), \dots, A_N^z = \text{Enc}_\varepsilon(\sigma_N^z)$, where z is a random value shared among the tallying authorities. The resulting *blinded* values σ_i^z are stored in a hash table for collision detection in linear time. Clearly, $\sigma_i = \sigma_j$ whenever $\sigma_i^z = \sigma_j^z$. In addition to eliminating duplicate votes, both authors propose using the same procedure for eliminating fake votes. In that case, however, based on the fact that the same exponent z is used across all ciphertexts, the coercer gets an attack strategy to identify whether a vote with known σ is counted [2, 9, 15]. Note that this attack does not apply to the elimination of duplicate votes.

Araujo et al. [1-4]. To solve the efficiency problem of the JCJ scheme, Araujo et al. suggest an approach based on group signatures. At registration, voters obtain their credentials. Unlike JCJ, no public values are related to voter roll entries. Their credentials enable the voters to deduce invalid credentials and mislead coercers. If the provided proofs hold, duplicates on the voting board are publicly identifiable by the equality of two values that are cast along with the vote. After mixing the relevant values on the voting board, the tallying authorities use their private keys to identify the legitimate votes. Notably, all information on their legitimacy is sent along with the vote itself, but can only be assessed by a sufficiently large group of tallying authorities. Fully avoiding plaintext equivalence tests between cast values and voter roll entries summarizes the essence of this elegant approach to avoid the inefficient comparison procedure.

An inherent weakness of this approach is the fact that a majority of colluding registrars could compute valid (but illegitimate) credentials unnoticed. As described earlier, adding illegitimate votes to the tally in JCJ requires the knowledge of a credential σ that complies with an entry S in the voter roll, i.e., such attacks could easily be detected. This is not the case in Araujo et al.'s scheme. Nevertheless, we believe that the approach holds much potential.

Spycher et al. [18]. This protocol strongly relates to the original JCJ protocol. For removing duplicates, they suggest using the linear-time scheme proposed by Smith and Weber. For fake vote elimination, they suggest preserving the use of the voter roll. The key to better efficiency lies in requiring voters to indicate which voter roll entry their vote (A, B) relates to. Tallying authorities then apply PET only on respective re-encryptions of A and S , where S is the public credential copied from the indicated voter roll entry. To preserve privacy, a certain number of fake votes is introduced by the authorities for each voter roll entry. This allows voters to deny the fact of having submitted their vote. Vote authorization thus becomes linear over the total number of submitted votes. More details on each steps of this protocol are given below.

Registration. The registration step is conducted according to JCJ. Additionally, it is assumed that a distinct public number, for example the index $i \in \{1, \dots, n\}$ of the voter's entry in the voter roll, is assigned to each voter.

Vote Casting. To cast a vote, the voters perform the same steps as in JCJ. In addition to the values A and B along with corresponding proofs, the value $C = \text{Enc}_\varepsilon(i, \alpha_C)$, accompanied by a NIZKP to prove knowledge of i , is posted to the bulletin board. The authorities later use i to locate the public credential S on the voter roll and thus to perform a single PET to efficiently eliminate fake votes. Note that the voting board must also accept encryptions C of values different from the voter's public number i .

Vote Authorization. After excluding votes with invalid proofs, the authorities add a random number β_i of additional fake votes for each voter (let β denote the average number of additional votes per voter). After duplicate elimination by applying Smith's and Weber's scheme on values A , the resulting adjusted list is passed as input to a first re-encryption mix-net, which outputs tuples (A', B', C') . Next, the authorities decrypt C' to extract i and establish a list of tuples (A', B', S) . Votes for which the decryption renders an invalid index $i \notin \{1, \dots, n\}$ are excluded from further processing. The remaining tuples are passed to a second re-encryption mix-net, which outputs tuples (A'', B'', S') . Now the tallying authorities perform $\text{PET}(A'', S')$ for each tuple. If the algorithm returns *false*, which is the case for all fake votes, the tuple is marked to be excluded from further processing.

Tallying. At the end, the tallying authorities jointly decrypt the values B'' of all remaining votes and publish the result.

Selections [6, 7]. Although SELECTIONS is based on JCJ, the approach presented by Clark and Hengartner has a slightly different setting. We will shortly summarize its phases and point out the differences.

Registration. The public credential is not an encryption of the voter's credential σ , but an encryption of g^σ for a publicly known generator g , i.e., $S = \text{Enc}_\varepsilon(g^\sigma, \alpha_S)$.

Election Setup. For every election, the public credentials are re-randomized into $S' = \text{Enc}_\varepsilon(\hat{g}^\sigma, \alpha_{S'})$, where $\hat{g} = g^\alpha$ is a fresh generator and $\alpha_{S'} = \alpha_S \cdot \alpha$ the new randomization for a random (but unknown) value α . The fresh generator \hat{g} is also used for casting votes. This mechanism prevents information leakage across elections.

Vote Casting. When casting a vote, the voter sends a commitment $A = \hat{g}^\sigma$, the encrypted vote $B = \text{Enc}_\varepsilon(c, \alpha_B)$, and a re-encryption of the public credential $C = \text{ReEnc}_\varepsilon(S', \alpha_C)$ to the public bulletin board. Additionally, an *anonymity set* containing S' and $\beta - 1$ randomly chosen public credentials different from S' is selected. Then the voter constructs a NIZKP to prove that C is a re-encryption of one of the β public credentials in the anonymity set. This proof together with a proof of knowledge of σ and a proof of well-formedness for $c \in \mathcal{C}$ are added to the ballot.

Vote Authorization. Similarly to JCJ, the tallying authorities first verify the proofs related to the submitted votes. If a proof does not hold for a vote, it is marked and excluded from further processing, i.e., invalid votes are eliminated. This can be performed during the vote casting phase, in particular at the moment the individual ballots arrive on the public bulletin board. The detection and elimination of duplicate votes follows from the simple fact that votes with the same credential will have the same commitment $A = \hat{g}^\sigma$. In that case, only one vote is kept for further processing (according to some policy). After eliminating duplicate votes, the remaining tuples (A, B, C) are mixed and re-encrypted (the commitment A is treated as an encryption with randomness 0) into (A', B', C') . For fake vote elimination, a simple PET between A' and C' is performed. If $\text{PET}(A', C')$ returns *false*, which is the case for all fake votes, the vote is marked and excluded from further processing.

Tallying. At the end, all remaining votes B' are jointly decrypted and tallied. The final result is published.

3 A New Protocol Based on Anonymity Sets

Similar to the protocols of Spycher et al. and SELECTIONS, our protocol strongly relates to the original JCJ protocol. For eliminating duplicate votes, we propose using the linear-time scheme of Smith and Weber, and for fake votes elimination, we suggest linking the vote to the voter roll. Instead of explicitly introducing fake votes by the authorities, we require the voter to specify an *anonymity set* of voters, similar to the one in SELECTIONS. The difference is that in our protocol, the voter only claims to be one of β voters without any proof for this claim. In contrast to SELECTIONS, the voter is thus not required to construct an expensive zero-knowledge proof during vote casting. In the vote authorization phase, the authorities replicate every submitted ballot into β ballots, one for each voter in the anonymity set. In other words, every submitted proper ballot leads to one authorized vote and $\beta - 1$ implicit fake votes. The protocol can therefore be regarded as a synthesis between SELECTIONS and the protocol of Spycher et al., where β , the size of the anonymity sets, constitutes an adjustable security parameter that determines the degree of coercion-resistance. More details on this idea are given in the subsequent description of the protocol. An overview of the protocol is depicted in Figure 2.

3.1 Protocol Description

In our description of the protocol, we follow the same style of exposition and notation as for the protocols described in Section 2. As many elements strongly relate to the original JCJ protocol and its successors, we will not explain everything again in comprehensive detail. Note that our protocol, in contrast with SELECTIONS, does not require a particular election setup phase.

Registration. The registration step is conducted according to Spycher et al. It is thus assumed that a distinct public number, for example the index $i \in \{1, \dots, n\}$ of the voter's entry in the voter roll, is assigned to each voter.

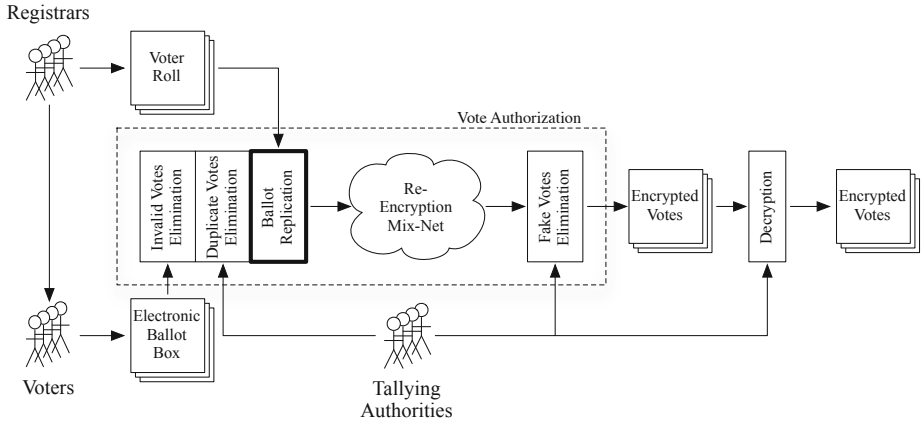


Fig. 2. Overview of the new protocol with some details about the vote authorization phase

Vote Casting. To cast a vote, the voter performs the same steps as in JCJ. Additionally to posting values A and B along with corresponding proofs, a subset $I \subseteq \{1, \dots, n\}$ of size β is chosen at random and added to the ballot. This is the ballot's anonymity set, which must include the voter's own index i to become a vote that counts. Since the voting board is a public bulletin board, the voter is able to individually verify that the vote has been cast as intended and recorded as cast.

Vote Authorization. In a first step, all votes with invalid proofs are marked to be excluded from further processing. Then duplicate votes are eliminated by applying Smith's and Weber's scheme on values A . Next, the authorities create β new ballots (A, B, S_j) for every submitted ballot and for every $j \in I$ in the corresponding anonymity set. These ballots are published on the voting board and thus, the correct construction is universally verifiable. The adjusted list of ballots is passed as input to a verifiable re-encryption mix-net, which outputs tuples (A', B', S'_j) . Now the authorities perform $\text{PET}(A', S'_j)$ for each tuple. All ballots for which the algorithm returns *false* are marked and excluded from further processing.

Tallying. The remaining unmarked ballots are authorized as legitimate votes and for every such ballot, B' is jointly decrypted and counted.

3.2 Security

Our protocol strongly relates to SELECTIONS and generally the same security considerations apply to our protocol. We do neither argue about the registration phase that corresponds to the original JCJ protocol nor about the final tallying phase, which essentially consists of decrypting the authorized ballots and counting them.² In the following,

² The security properties of the tallying phase depend on the encryption scheme that is used and is not related to the security properties of our protocol (as long as the encryption scheme allows verifiable mixing and does not allow the coercer to link the cast vote to the mixed ballots).

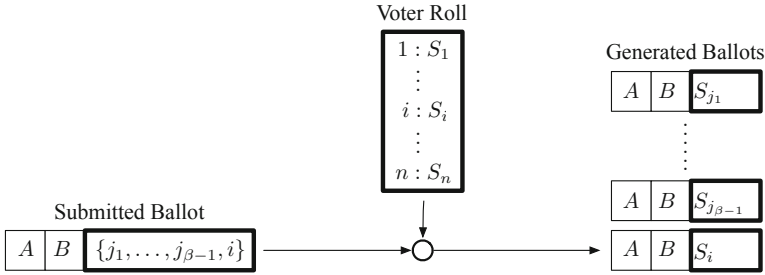


Fig. 3. The ballot replication step for one submitted ballot

we will briefly explain how vote casting and vote authorization in our protocol relates to SELECTIONS and recap the security considerations of SELECTIONS with respect to the security parameter β .

Vote Casting. As it is the case for every coercion-resistant voting protocol, the voter must have one moment of privacy to cast the real vote. Hence, the coercer can only observe the public bulletin board to check for the voter's compliance. During the vote casting phase, the public bulletin board reveals the same information to a coercer as it is the case for SELECTIONS. Particularly, the coercer knows the number of votes associated with each voter. The greater the security parameter β , the more likely the coerced voter's index occurs in someone else's anonymity set and thus the less certain the coercer will be regarding the voter's compliance.

Vote Authorization. The coercer does not get additional information on whether the vote will be tallied or not from observing the elimination of invalid votes. The next step, the elimination of duplicate votes using the technique of Smith and Weber is the same as in the scheme of Spycher et al. and is similar to SELECTIONS, i.e., the same security considerations apply. The ballot replication step is different to all existing approaches. However, the straightforward inflation of the submitted ballots with the indices of the anonymity set is public and does obviously not reveal any additional information to the coercer. The last step of the vote authorization phase, the mixing of the remaining ballots and the elimination of fake votes, is again the same as in Spycher et al. and SELECTIONS, i.e., the same security arguments apply here. In particular, because of the fact that the coercer cannot link the eliminated ballots to the initial votes, no information is leaked to the coercer on whether the coerced voter's fake vote was finally accepted or not.

Security Parameter β . Clark and Hengartner provide a thorough analysis regarding the security parameter β (size of the anonymity sets) in SELECTIONS [7]. The exact same considerations can be applied for our approach. There are essentially three interesting cases:

1. The anonymity set includes the entire voter roll, that is $\beta = n$. In this case, the degree of coercion-resistance corresponds to JCJ, but vote authorization (or more precisely fake vote elimination) is again quadratic in n (or more precisely multilinear in n and N , see Table 3).
2. The anonymity set has a fixed size, for example $\beta = 50$. Clark and Hengartner showed that a coercer may decide with small but non-negligible probability whether or not the coerced voter complied with the instructions. Hence, if β is a constant value, coercion-resistance is not given to the full, but depending on the value of β , to a reasonable extent.
3. The size of the anonymity set varies among voters, but has a minimal size $\beta_i \geq \beta$, for example $\beta_i \geq 50$. Clark and Hengartner point out the possibility of coerced voters to place their real vote as *stealth votes* with $\beta_i = n$. They also emphasize that in this situation, they need to assume that other stealth votes are cast as well to assure *adversarial uncertainty*.

Temporal Aspects. The security experiments of JCJ do not fully capture some important temporal aspects. For example, it is assumed that all honest voters submit their ballots in only one step (i.e., in parallel). In a more realistic setting, the coercer may observe the order and time when the ballots arrive on the public bulletin board. Consider for example the case where the coerced voter has only a short moment of privacy during night time, when only few other voters are casting votes. Observing the public bulletin board during this time, the coercer might get a strong indication whether or not the coerced voter really complied. The problem is especially problematical in both SELECTIONS and our approach, since the probability that the coerced voter appears in another voter's anonymity set might be low (depending on the actual choice of the security parameter β). To counteract the corresponding advantage for the coercer, we propose the following extension to our approach:

1. Before casting the ballot, the voter encrypts the anonymity set with the public key of the tallying authorities.
2. After eliminating duplicates, but before ballot replication, the ballots are mixed and re-encrypted in an additional re-encryption mix-net (similar to Spycher et al.).
3. The tallying authorities jointly decrypt the anonymity sets included in the mixed ballots.

Formal Proofs. Coercion-resistance of our protocol can be proved under the game-based definition of Juels et al. [13]. Since our approach is close to the approach presented by Clark and Hengartner, their security games serve as a starting point for the formal proofs, which we will carry out as future work.

4 Performance Comparison

This section is dedicated to a performance comparison of all the above introduced approaches. Our comparison excludes the one-time registration phase. In our results, the number of registered voters is denoted by n , the number of submitted ballot by N , the number of mixing and tallying authorities by T , the number of candidates by m , and

the size of the anonymity sets by β . We take the work of Clark and Hengartner [6, 7] as a starting point and augment their findings with the performance properties of our protocol and the one of Spycher et al. We make similar assumptions to facilitate a better comparison:

- We only use standard ElGamal encryption over a modular multiplicative group of integers (i.e., not the modified ElGamal version of JCJ).
- We only count the number of necessary modular exponentiations to perform the respective tasks (i.e., all other arithmetic operations are neglected).
- We do not use techniques, which could equally improve the performance of all protocols (e.g., the “blocking technique” of CIVITAS).
- We assume that a valid vote consists of exactly one candidate $c \in \mathcal{C}$, where $m = |\mathcal{C}|$ denotes number of candidates.
- We assume that the re-encryption mix-nets use *randomized partial checking* [12] for proving the correctness of the mixing (i.e., each authority mixes the encryptions twice and half of these re-encryptions are checked).
- We assume that all encrypted votes are decrypted during the tallying phase (i.e., no homomorphic tallying).
- We assume that all tallying authorities participate at the vote authorization phase (e.g., distributed instead of threshold decryption or PET).
- We assume that all commitments in the distributed operations are based on hash functions.

In contrast to Clark and Hengartner, we include the proofs of well-formedness of the encrypted votes in our calculations. We also take the election setup of their protocol and the tallying phase into consideration. For improving the readability of the results, we have re-arranged the values for checking the proofs in a separate *verification phase*. To simplify the results given in [6, 7], we consider only the worst case when all submitted ballots reach the fake vote elimination phase. In other words, we assume that all submitted ballots contain valid proofs and that the ballot box contains no duplicate votes (but searching for invalid and duplicate votes is still necessary). We also assume that every registered voter has submitted at least one valid vote, i.e., the number of votes to decrypt during tallying is exactly n and $N \geq n$.

4.1 Performance Analysis

Table 2 and Table 3 summarize the results of the performance analysis. Table 2 is based on corresponding values for the cryptographic primitives as given in Table 1 and Table 3 shows the same results more compactly in Big-Oh notation. Note that some of the values in Table 1 slightly differ from the ones given in [6, 7]. Since proving knowledge of a plaintext requires only a proof of knowledge of the encryption randomness (Schnorr), it can be constructed with 1 and verified with 2 exponentiations. Proving the correct decryption corresponds to proving a single equality of discrete logarithms (Chaum-Pederson). In the distributed case with T authorities, this simply scales up to T , $2T$, and $4T$ exponentiations for performing the partial decryptions, constructing the proofs, and verifying the proofs, respectively. Finally, a proof of encrypting 1-out-of- m plaintexts requires $4m-2$ exponentiations to construct (2 for the correct value and 4 for

Table 1. Number of necessary exponentiations for various cryptographic primitives according to the procedures as described in [3, 8, 16]

	Perform Operation	Generate Proof(s)	Verify Proof(s)
<i>Encryption:</i> Standard ElGamal encryption with proof of knowledge of plaintext (Schnorr).	2	1	2
<i>Well-Formed Encryption:</i> Standard ElGamal encryption of 1-out-of- m possible plaintexts with proof of well-formedness (Chaum-Pederson, OR-composition).	2	$4m - 2$	$4m$
<i>Re-Encryption:</i> Standard ElGamal re-encryption with proof of correctness (Chaum-Pederson).	2	2	4
<i>Well-Formed Re-Encryption:</i> Standard ElGamal re-encryption of 1-out-of- β possible encryptions with proof of well-formedness (Chaum-Pederson).	2	$4\beta - 2$	4β
<i>Mixing:</i> Re-encryption and permutation of N ciphertext tuples of length k .	$2kN$	$2kN$	$4kN$
<i>Mix-Net:</i> T authorities perform a re-encryption mix-net with randomized partial checking (double re-encryption, but only half of the proofs are provided).	$4kNT$	$2kNT$	$4kNT$
<i>Commitment:</i> Applying an exponent with proof of knowledge (Schnorr).	1	1	2
<i>Distributed Commitment:</i> T authorities applying exponents with proofs of knowledge (Schnorr).	T	T	$2T$
<i>Blinding:</i> Applying an exponent on the ciphertext with proof of correctness (Chaum-Pederson).	2	2	4
<i>Distributed Blinding:</i> T trustees applying exponents on the ciphertexts with proofs of correctness (Chaum-Pederson).	$2T$	$2T$	$4T$
<i>Decryption:</i> Standard ElGamal decryption with proof of correctness (Chaum-Pederson).	1	2	4
<i>Distributed Decryption:</i> T trustees performing distributed ElGamal decryptions with proofs of correctness (Chaum-Pederson).	T	$2T$	$4T$
<i>Plaintext Equivalence Test:</i> Distributed blinding followed by distributed decryption.	$3T$	$4T$	$8T$

every $m-1$ simulated values in the OR-composition) and $4m$ exponentiations to verify. Similarly, proving the re-encryption of 1-out-of- β ciphertexts requires $4\beta-2$ exponentiations to construct and 4β to verify the proof. The results given in Table 2 are based on these modifications, but they have no impact on the asymptotic results of Table 3. Note that the performance of the final tallying by decrypting the valid votes in a distributed way is the same in all protocols (nT exponentiations for performing the decryptions, $2nT$ for constructing the proofs, and $4nT$ for verifying the proofs). More details about the performance calculations are given in the upcoming paragraphs.

a) *JCJ (CIVITAS)*. Vote casting consist of two encryptions ($= 4$) with one proof of knowledge of plaintext ($= 1$) and one proof of well-formedness ($= 4m-2$). Vote authorization requires verifying N proofs of knowledge ($= 2N$) and N proofs of well-formedness ($= 4mN$) to eliminate invalid votes, $\binom{N}{2}$ many PETs ($= \frac{3}{2}(N^2-N)T$)

Table 2. Performance comparison by counting the number of modular exponentiations required in each phase

	JCJ (CIVITAS)	Araujo et al.	Spycher et al.	Clark et al. (SELECTIONS)	Our Protocol
<i>Election Setup</i>	–	–	–	$(4n+2)T$	–
<i>Vote Casting</i>	$4m+3$	$4m+13$	$4m+6$	$4\beta+4m+2$	$4m+3$
<i>Vote Authorization</i>					
Eliminate Invalid Votes	$(4m+2)N$	$(4m+10)N$	$(4m+4)N$	$(4\beta+4m+2)N$	$(4m+2)N$
Insert Fake Votes	–	–	$6\beta n$	–	–
Elim. Duplicate Votes	$\frac{7}{2}(N^2-N)T$	0	$7(N+\beta n)T$	0	$7NT$
1st Mixing of Ballots	$12NT$	$30NT$	$18(N+\beta n)T$	$18NT$	$18\beta NT$
2nd Mixing of Ballots	–	–	$21(N+\beta n)T$	–	–
Mixing of Credentials	$6NT$	–	–	–	–
Eliminate Fake Votes	$7nNT$	$(14T+6)N$	$7(N+\beta n)T$	$7NT$	$7\beta NT$
<i>Tallying</i>	$3nT$	$3nT$	$3nT$	$3nT$	$3nT$
<i>Verification</i>					
<i>Election Setup</i>	–	–	–	$4(n+1)T$	–
Eliminate Invalid Votes	$(4m+2)N$	$(4m+10)N$	$(4m+4)N$	$(4\beta+4m+2)N$	$(4m+2)N$
Elim. Duplicate Votes	$4(N^2-N)T$	0	$8(N+\beta n)T$	0	$8NT$
1st Mixing of Ballots	$8NT$	$20NT$	$12(N+\beta n)T$	$12NT$	$12\beta NT$
2nd Mixing of Ballots	–	–	$16(N+\beta n)T$	–	–
Mixing of Credentials	$4NT$	–	–	–	–
Eliminate Fake Votes	$8nNT$	$(16T+8)N$	$8(N+\beta n)T$	$8NT$	$8\beta NT$
<i>Tallying</i>	$4nT$	$4nT$	$4nT$	$4nT$	$4nT$

with proofs ($= 2(N^2-N)T$) to eliminate duplicates, a re-encryption mix-net for N encryption pairs ($= 8NT$) with proofs ($= 4NT$) to mix the ballots, a second re-encryption mix-net for n single encryptions ($= 4NT$) with proofs ($= 2NT$) to mix the credentials, and finally nN many PETs ($= 3nNT$) with proofs ($= 4nNT$) to eliminate fake votes. Corresponding values for verifying the proofs follow accordingly.

b) Araujo et al. We use the latest version of the protocol for the comparison [3] and adopt the analysis provided in [7]. Vote casting consists of four encryptions ($= 8$) with three proofs of knowledge of plaintext ($= 3$) and one proof of well-formedness ($= 4m - 2$).

Table 3. Performance comparison by describing the asymptotic growth of the numbers of exponentiations in each phase using the Big-Oh notation (relative to n , N , β , m , and T)

	JCJ (CIVITAS)	Araujo et al.	Spycher et al.	Clark et al. (SELECTIONS)	Our Protocol
<i>Election Setup</i>	–	–	–	nT	–
<i>Vote Casting</i>	m	m	m	$\beta+m$	m
<i>Vote Authorization</i>	$N^2T+nNT+mN$	$NT+mN$	$NT+\beta nT+mN$	$NT+\beta N+mN$	$\beta NT+mN$
$T, m = \text{const.}$	N^2+nN	N	$N+\beta n$	βN	βN
<i>Tallying</i>	nT	nT	nT	nT	nT
<i>Verification</i>	$N^2T+nNT+mN$	$NT+nT+mN$	$NT+\beta nT+mN$	$NT+\beta N+nT+mN$	$\beta NT+nT+mN$
$T, m = \text{const.}$	N^2+nN	$N+n$	$N+\beta n$	$\beta N + n$	$\beta N + n$

Two of the encrypted values and one of the non-encrypted values need one exponentiation to compute ($= 3$). A proof of representation that relates the non-encrypted to one of the encrypted values ($= 1$) is added to the ballot. To eliminate invalid votes, vote authorization requires verifying $3N$ proofs of knowledge ($= 6N$), N proofs of well-formedness ($= 4mN$), and N proofs of representation ($= 4N$). Duplicates can be removed at no additional costs. As suggested in [7], we omit the additional encryption step, which can be performed by the first mix-net authority. The re-encryption mix-net takes N encryption 5-tuples ($= 30NT$) as input and produces corresponding proofs ($= 20NT$). Finally, eliminating fake votes requires for each vote two commitments ($= 2N$), two Chaum-Pederson proofs ($= 4N$), and two PETs ($= 6NT$) with proofs ($= 8NT$). Corresponding values for verifying the proofs follow accordingly.

c) Spycher et al. Vote casting consist of three encryptions ($= 6$) with two proofs of knowledge of plaintext ($= 2$) and one proof of well-formedness ($= 4m-2$). Vote authorization requires verifying $2N$ proofs of knowledge ($= 4N$) and N proofs of well-formedness ($= 4mN$) to eliminate invalid votes, three encryptions without proofs for each of the βn inserted fake votes ($= 6\beta n$), $N+\beta n$ many distributed blinding operations ($= 2(N+\beta n)T$) with proofs ($= 2(N+\beta n)T$) and distributed decryptions ($= (N+\beta n)T$) with proofs ($= 2(N+\beta n)T$) to eliminate duplicates (Smith's and Weber's scheme), a re-encryption mix-net for $N+\beta n$ encryption triples ($= 12(N+\beta n)T$) with proofs ($= 6(N+\beta n)T$) to mix the ballots, $N+\beta n$ distributed decryptions ($= (N+\beta n)T$) with proofs $2(N+\beta n)T$ plus another re-encryption mix-net for $N+\beta n$ encryption triples ($= 12(N+\beta n)T$) with proofs ($= 6(N+\beta n)T$) to mix the ballots a second time, and finally $N+\beta n$ many PETs ($= 3(N+\beta n)T$) with proofs ($= 4(N+\beta n)T$) to eliminate fake votes. Corresponding values for verifying the proofs follow accordingly.

d) Clark et al. (SELECTIONS). The election setup requires n distributed blinding operations ($= 2nT$), one distributed commitment ($= T$) and an AND-composition of corresponding proofs ($= 2nT + T$). Vote casting consist of a commitment ($= 1$) with a proof of knowledge ($= 1$), a re-encryption ($= 2$) with a proof of well-formedness ($= 4\beta - 2$), and one encryption ($= 2$) with a proof of well-formedness ($= 4m - 2$). Vote authorization requires a re-encryption mix-net for N encryption triples ($= 12NT$) with proofs ($= 6NT$) to mix the ballots, and finally N many PETs ($= 3NT$) with proofs ($= 4NT$) to eliminate fake votes. Corresponding values for verifying the proofs follow accordingly.

e) Our Protocol. Vote casting consist of two encryptions ($= 4$) with one proof of knowledge of plaintext ($= 1$) and one proof of well-formedness ($= 4m - 2$). Vote authorization requires verifying N proofs of knowledge ($= 2N$) and N proofs of well-formedness ($= 4mN$) to eliminate invalid votes, N many distributed blinding operations ($= 2NT$) with proofs ($= 2NT$) and distributed decryptions ($= NT$) with proofs ($= 2NT$) to eliminate duplicates (Smith’s and Weber’s scheme), a re-encryption mix-net for βN encryption triples ($= 12\beta NT$) with proofs ($= 6\beta NT$) to mix the ballots, and finally βN many PETs ($= 3\beta NT$) with proofs ($= 4\beta NT$). Corresponding values for verifying the proofs follow accordingly.

4.2 Discussion

In our new protocol, casting a vote is as efficient as in the original JCJ protocol or in CIVITAS. For a fixed candidate set, a constant number of exponentiations is needed. If the candidate set is reasonably small, this seems to be feasible on today’s typical client platforms (e.g., 11 exponentiations are needed for $m = 2$ choices in a referendum). Compared to SELECTIONS, where casting a vote depends on the security parameter β , this is the main advantage of our approach. We think that for reasonably large anonymity sets, the protocol of Clark et al. is not competitive enough to be considered as a solution for a coercion-resistant voting system (e.g., 210 exponentiations are needed for $\beta = 50$ and $m = 2$). Asymptotically, the number of exponentiations is $O(\beta + m)$ for SELECTIONS and $O(m)$ for all the others (see Table 3).

To determine the protocol with the most efficient vote authorization procedure, for example by interpreting the general asymptotic results in Table 3 we need to take into account multiple systems parameters. To facilitate this task, we propose two simplifications: we consider a constant number of authorities and a fixed candidate set (both T and m affect all protocols in a similar way). The corresponding simplified growth rates are shown in Table 3 below the general results. While JCJ and CIVITAS are essentially quadratic in N , it turns out that Araujo et al.’s protocol—although it has relatively high constant factors—is the only one that is truly linear in N . Among the others, the advantage of Spycher et al.’s protocol is the fact that β only multiplies with n , the number of voters, which is fixed for a given election (whereas the number of submitted ballots N has no upper bound). On the other hand, Spycher et al.’s protocol has the least favorable constant factors among all. Our new protocol and SELECTIONS are comparable with respect to their growth rates, but SELECTIONS has a significantly lower constant

factor for βN . However, SELECTIONS allows to carry out the invalid votes elimination already during the vote casting phase, which is a considerable advantage compared to our protocol. Note that the same conclusions hold for the verification procedure.

5 Conclusion

We have presented a new improvement of the JCJ protocol that allows efficient vote authorization without requiring more computation power on the voter's side. Conceptually, it is a mix between the existing protocols of Spycher et al. and SELECTIONS. To conclude this paper, we summarize the phases with emphasis on vote authorization and compare our protocol with the different approaches discussed in this paper.

Election Setup. In contrast to SELECTIONS, our protocol as well as the other examined protocols require no election setup.

Vote Casting. Compared with the protocol of Spycher et al. we do not require the authorities to generate random fake votes during the vote casting phase and therefore we reduce the effort for the authorities in this phase. In contrast to SELECTIONS, our improvement introduces no additional effort for the voter in terms of modular exponentiations. The voter only has to add a set of β voter roll indices to the ballot. When applied on systems and technologies with limited computing resources such as mobile phones or web applications using JavaScript, all additional performance requirements are undesirable. Another advantage of our approach is the fact, that security parameter β does not affect the client-side at all. We believe that security should not be bounded by the computing equipment of the individual voters or even require them to buy better computers to protect their privacy or an e-voting protocol to a reasonable degree. Moreover, this contradicts the fundamental principle of equality. In our approach β only affects the server-side performance requirements which is more scalable with respect to computation power.

Vote Authorization. The lower computation requirements for the voters during the vote casting phase yield more effort to put in the vote authorization phase. Security parameter β affects the computational requirements on the server-side as a linear factor. In particular, we need to explicitly remove duplicates using the linear approach of Smith and Weber and we enlarge the input of the mix-net by factor β . Hence, mixing in our protocol requires additional computing power compared to the other protocols.

Current and Future Work. Since our protocol strongly relates to existing proven concepts, we informally justified the correctness of the individual phases. However, future work includes formal proofs of correctness of these arguments. Currently, we are engaged in developing prototypes for various coercion-resistant voting protocols. Our experience with these realizations will allow us to compare the existing approaches from a more practical perspective.

Acknowledgments. We thank Jeremy Clark and the three anonymous reviewers for their constructive comments. This research is supported by the *Swiss Federal Chancellery*, the *Hasler Foundation* (project No. 09037), and the *Mittelbauförderung* of the Bern University of Applied Sciences.

References

1. Araujo, R.: On Remote and Voter-Verifiable Voting. PhD thesis, Department of Computer Science, Darmstadt University of Technology, Darmstadt, Germany (2008)
2. Araújo, R., Foulle, S., Traoré, J.: A practical and secure coercion-resistant scheme for remote elections. In: Chaum, D., Kutyłowski, M., Rivest, R.L., Ryan, P.Y.A. (eds.) FEE 2007, Frontiers of Electronic Voting, Schloss Dagstuhl, Germany, pp. 330–342 (2007)
3. Araújo, R., Foulle, S., Traoré, J.: A Practical and Secure Coercion-Resistant Scheme for Internet Voting. In: Chaum, D., Jakobsson, M., Rivest, R.L., Ryan, P.Y.A., Benaloh, J., Kutyłowski, M., Adida, B. (eds.) Towards Trustworthy Elections. LNCS, vol. 6000, pp. 330–342. Springer, Heidelberg (2010)
4. Araújo, R., Ben Rajeb, N., Robbana, R., Traoré, J., Youssfi, S.: Towards Practical and Secure Coercion-Resistant Electronic Elections. In: Heng, S.-H., Wright, R.N., Goi, B.-M. (eds.) CANS 2010. LNCS, vol. 6467, pp. 278–297. Springer, Heidelberg (2010)
5. Brandt, F.: Efficient Cryptographic Protocol Design Based on Distributed El Gamal Encryption. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 32–47. Springer, Heidelberg (2006)
6. Clark, J.: Democracy Enhancing Technologies: Toward Deployable and Incoercible E2E Elections. PhD thesis, University of Waterloo, Canada (2011)
7. Clark, J., Hengartner, U.: Selections: Internet Voting with Over-the-Shoulder Coercion-Resistance. In: Danezis, G. (ed.) FC 2011. LNCS, vol. 7035, pp. 47–61. Springer, Heidelberg (2012)
8. Clarkson, M.R., Chong, S., Myers, A.C.: Civitas: Toward a secure voting system. Technical Report TR 2007-2081, Department of Computer Science. Cornell University (2007)
9. Clarkson, M.R., Chong, S., Myers, A.C.: Civitas: Toward a secure voting system. In: SP 2008, 29th IEEE Symposium on Security and Privacy, Oakland, USA, pp. 354–368 (2008)
10. Di Cosmo, R.: On privacy and anonymity in electronic and non electronic voting: the ballot-as-signature attack. Hyper Articles en Ligne, hal-00142440(2) (2007)
11. 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)
12. Jakobsson, M., Juels, A., Rivest, R.L.: Making mix nets robust for electronic voting by randomized partial checking. In: Boneh, D. (ed.) SS 2002, 11th USENIX Security Symposium, San Francisco, USA, pp. 339–353 (2002)
13. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: Atluri, V., De Capitani di Vimercati, S., Dingledine, R. (eds.) WPES 2005, 4th ACM Workshop on Privacy in the Electronic Society, Alexandria, USA, pp. 61–70 (2005)
14. Koenig, R., Haenni, R., Fischli, S.: Preventing Board Flooding Attacks in Coercion-Resistant Electronic Voting Schemes. In: Camenisch, J., Fischer-Hübner, S., Murayama, Y., Portmann, A., Rieder, C. (eds.) SEC 2011. IFIP AICT, vol. 354, pp. 116–127. Springer, Heidelberg (2011)
15. Pfitzmann, B.: Breaking an Efficient Anonymous Channel. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 332–340. Springer, Heidelberg (1995)

16. Rjasková, Z.: Electronic voting schemes. Diploma thesis, Department of Computer Science. Comenius University, Bratislava, Slovak Republic (2002)
17. Smith, W.D.: New cryptographic voting scheme with best-known theoretical properties. In: FEE 2005, Workshop on Frontiers in Electronic Elections, Milan, Italy (2005)
18. Spycher, O., Koenig, R., Haenni, R., Schläpfer, M.: A New Approach towards Coercion-Resistant Remote E-Voting in Linear Time. In: Danezis, G. (ed.) FC 2011. LNCS, vol. 7035, pp. 182–189. Springer, Heidelberg (2012)
19. Weber, G., Araujo, R., Buchmann, J.: On coercion-resistant electronic elections with linear work. In: ARES 2007, 2nd International Conference on Availability, Reliability and Security, Vienna, Austria, pp. 908–916 (2007)
20. Weber, S.: Coercion-Resistant Cryptographic Voting: Implementing Free and Secret Electronic Elections. VDM Verlag, Saarbrücken (2008)

The Bug That Made Me President a Browser- and Web-Security Case Study on Helios Voting

Mario Heiderich, Tilman Frosch, Marcus Niemiets, and Jörg Schwenk

Chair for Network and Data Security
Ruhr-University Bochum, Germany

Abstract. This paper briefly describes security challenges for critical web applications such as the Helios Voting system. After analyzing the Helios demonstration website we discovered several small flaws that can have a large security critical impact. An attacker is able to extract sensitive information, manipulate voting results, and modify the displayed information of Helios without any deep technical knowledge or laboratory-like prerequisites. Displaying and processing trusted information in an untrustworthy user agent can lead to the issue that most protection mechanisms are useless. In our approach of attacking Helios voting systems we do not rely on an already infected or trojanized machine of the user, instead we use simple and commonly known web browser features to leverage information disclosure and state modification attacks. We propose that online voting applications should at least follow the latest vulnerability mitigation guidelines. In addition, there should be thorough and frequent coverage with automated as well as manual penetrations tests in privacy sensitive applications. E-Voting software driven by web browsers is likely to become an attractive target for attackers. Successful exploitation can have impact ranging from large scale personal information leakage, financial damage, calamitously intended information and state modification as well as severe real life impact in many regards.

Keywords: Web Application Security, Privacy, E-Voting, Browser Security, Vulnerability Mitigation, Information Disclosure.

1 Introduction

Helios is an online voting web application framework composed in Python. The core is fully open sourced, can be reviewed and forked on Github and is currently being maintained by B. Adida, O. de Marneffe, and O. Pereira [1]. Helios was peer-reviewed in 2008 [2]. It makes use of client-side data encryption to sustain its security model. Furthermore, it ensures that the surveyed ballot data can securely traverse the communication layers from user agent to verification instances and voting servers. Since version three, Helios does not exclusively rely on Java Applets and LiveConnect [3] for client side encryption anymore. Instead Helios makes use of a JavaScript-based encryption library. However, fully JavaScript-based client side cryptography is currently being implemented, but not deployed

in public versions so far. Still parts of the application do not work properly in case that a user has deactivated JavaScript or uses a restricted configuration.

The currently available demonstration website¹ allows users to log in via Twitter, Google, Facebook, or Yahoo and to quickly set up elections containing different questions with different sets of possible answers. Helios also allows to have voters publish their voting results with the help of social networking features by interacting with Twitter and Facebook APIs. Helios is fully open sourced so that anyone who is questioning its inner workings can have a closer look into the source code by analyzing the Python and JavaScript files of the system. Another benefit of Helios is being *open-audit* - which means that even without a source code audit or external validation mechanisms the election data can be verified based on individual ballot encryption – adding another layer of security and trustability.

The International Association for Cryptologic Research (IACR) initiated an electronic election demonstration in August 2009 [4]. As a consequence they used Helios as voting system later on to get feedback from the IACR members. The results were inter alia that over ninety-one percent want to switch to electronic voting over the Internet for IACR decisions. In addition, over eighty-eight percent would like to use Helios as voting system. The acceptance of an electronic voting system and, especially, of Helios is extremely high. This underlines the importance to conduct a security analysis of this tool.

Risk Assessment. A serious risk for Helios-based applications and voting processes are client side attack techniques and patterns such as Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), or remote state detection and manipulation as well as phishing attacks [5]. The cryptographic protocol and the security mechanisms protecting the voting process and the vote transmission become meaningless as soon as an attacker is able to inject client side JavaScript or even plugin code, for the reason that he or she is able to modify the data the voter is working on – or record and extract keystrokes, clicks, and mouse movements to unveil the voters actions and voting intents.

Whereas after a short test no “low hanging fruit” XSS vulnerabilities were spotted, Helios nevertheless suffers several feature-based weak spots – including more complex XSS problems an attacker might initially overlook. They allow for an attacker to easily interfere with the voting process by manipulating data or by redirecting users to different domains providing malicious lookalike pages of the voting application. Attacker-controlled malicious web pages offer possibilities for stealing and back-channeling voting results and other sensitive information. This can be done by using user interface (UI)-based attack techniques such as Clickjacking [6], Drag & Drop-based attacks [7], which can lead to content extraction, and other recently published techniques.

Attacks. In section 2 we elaborate on some simple yet effective web attacks against Helios-based voting applications. We show that simple measures are sufficient to interrupt or heavily disturb the voting process. These measures lead

¹ <https://vote.heliosvoting.org>

to a situation, where it is at least feasible to leak private information. We also present several trivial but effective attacks against Helios by utilizing a Document Object Model (DOM) property, which is used to name frames and identify window handles for redirects and cross-domain location changes. Furthermore we list and discuss further design and implementation bugs we discovered in Helios, including CSRF problems, XSS bugs and additional ways to attack the platform and extract sensitive data.

We demonstrate how an attacker is able to use small and seemingly harmless features and tricks to generate a compound of attacks capable of completely compromising a Helios-based voting application. Unlike Esteghahri et al. [8] we actually found existing client side vulnerabilities in Helios and didn't require the user to open a PDF or any other malicious file to successfully compromise an election. We will further point out the dangers as well as the risks of web-based voting systems and survey software. In addition, we provide mitigation concepts and suggestions how to remedy the vulnerabilities we discovered. With the fixing instruction we provide, we hope to improve Helios one step further to be a robust and secure web-based voting system, to make it capable to cope with the immane tasks such a software is entrusted with.

2 Exploits and Vulnerabilities

One of the most effective ways to interfere with the voting process of a possible victim is abusing a window handle the Helios application is connected with. In case an attacker can spawn or even just *link* a Helios instance with a name that he or she controls, there is an easy way to remote control the handle's location – even across domain borders. This allows an attacker to interfere with an election process, leak sensitive data and redirect a user to a different website in the middle of a voting step without him necessarily noticing this dangerous interruption. Among other reasons we will point out, this is due to the fact that the public Helios demonstration application neither uses HTTP headers, which check if a browser is allowed to render the web page in a frame or not, nor JavaScript frame busting code performing this task. Neither does the Helios application reset the window's name to prevent any UI redressing attack. Thus, an attacker can control the tab, frame, or pop-up's location string. We discuss the implication below.

2.1 Attacking Named Windows

Consider the following attack case: during the time of a public voting an attacker sends out many phishing mails inviting a user to the real and unmodified voting application. The user will access a website, crafted by the attacker, where he finds a link to the actual benign voting website. Alternatively the attacker will use a HTML mail already containing the link to the election website. Note: In this scenario nothing is wrong with the linked target – it is the actual application

without having any suspicious parameters attached. No modifications have maliciously taken place with the voting application before. The attacker amends the link pointing to the voting application with a `target` attribute. This attribute ensures that the `window.name` variable of the resulting new tab or window will be pre-set to the exact value of the target attribute. The example code shown in Listing 1.1 demonstrates that.

```
<a target="public_handle"
href="https://benign.voting.website/">
    Voting Application
</a>
```

Listing 1.1. Malicious website linking the voting application

The attacker can now interact with the freshly created frame, tab, or pop-up window and change its URL without any restrictions. This is caused by the frame's Document Object Model (DOM) property `window.name`; it will be assigned to the value `public_handle` and any other clicked link or call on `window.open()` bearing this target will cause a location change on the voting application tab.

```
<a href="#" onclick="open('/evil.com', 'public_handle');
return false;">
    Voting Application
</a>
```

Listing 1.2. The `window.open()` method addressing a specific frame handle

The only limitation in this context are Cross Site Scripting (XSS) attempts to set the target's URL to a JavaScript URI and thereby executing JavaScript cross domain and stealing sensitive data directly from the attacked DOM. This is not working in modern browsers as our tests showed. However, the attacker can still use timing attacks to find out, which state of the voting process the victim is currently residing in and thus deploy specific and visually matching arbitrary content by changing the URL in the correct moment. The only necessary requirement for this is to have an active tab, window, or even pop-under, in which the attacker can execute JavaScript over a long period of time. If the user closes this browser instance the attack will not work anymore – unless the attacker spawns a new tab or window in case of an `onunload` event.

More specific UI attacks are possible based on an unset or guessable `window.name` variable. Nevertheless, this example shows the true potential quite well. Few to no suspicious user interactions are necessary to be able to read sensitive data, redirect the user to a different but similar looking application on a different domain and retrieve passwords and other sensitive information. Clickjacking attacks can be considered even more dangerous in this case, since an attacker only has to take different frames of the same voting application and overlap them with each other to trick the user into selecting choices and casting

a ballot for arbitrary candidates – which might possibly differ from the original voting intention [9]. The Tab-Nabbing attacks described by Raskin [10][11] relate to these kinds of tricks as well, same as the CSS-based injection attacks published by Barth et al [12], discussed in subsection 2.3.

During our tests against the public Helios demonstration website we also noticed a lack of security measures to protect the DOM against an attacker who is capable to execute JavaScript on this domain. There are several ways to limit the possibilities for post exploitation in case of an XSS attack. This includes JavaScript sandboxes, proper Content Security Policy (CSP) headers as well as a good architectural style chosen for the websites client side business logic [13]. Almost the whole client side business logic relies on global variables easily accessible to the attacker, which includes the whole cryptographic logic. The attacker can – in case that JavaScript code was injected successfully – easily overwrite and thus tap global methods such as `save_questions()` or `str_hmac_sha1()`.

A first aid approach to fix the global leakage of important and critical DOM methods would be to wrap the whole business logic into an anonymous JavaScript method – this would prevent easy *DOM clobbering* and method overwriting [14]. Furthermore, a JavaScript sand-boxing approach should be chosen to make sure an attacker cannot execute arbitrary client side script code without having to bypass the sandbox restrictions [15]. Instrumentation of a library such as the OWASP ESAPI for JavaScript could help encoding user generated data properly, in case it's being handled by the JavaScript layer only, to prevent DOMXSS [16]. It's worth to note though: none of the currently available JavaScript sandboxes provide reliable and tamper-proof security on all modern browsers. Nevertheless a JavaScript sandbox is one of many ways to raise the bar in terms of web application security. An application, which is as critical as a voting tool, should include any available countermeasure to provide a higher degree of security.

2.2 Cross-Site Scripting and Open Redirects

The Helios system suffers a vulnerability in the administration area allowing an attacker to create an open redirect. The URL parameter `continue_url` offers the possibility to place malicious code into the web page via an `a`-tag, which can be influenced by an attacker to redirect the administrator to an external domain without necessarily noticing the domain change. The attack scenario is surprisingly simple to set up – nothing more has to be done than to trick the election administrator into clicking a malicious link from arbitrary sources. These can include emails, instant messaging (IM), or even documents such as DOC files and PDFs. The attacker does not have to know any secret data such as a URL token representing the unique ID of the election - the URL is fully generic. This attack enables effective and simple phishing attempts and can easily be carried out even by non-tech-savvy adversaries.

To escalate the whole attack and therefore accomplish a fully reflected script execution on the actual domain where the election is being stored, an attacker can utilize a self-contained URI resource, also known as data URI [17]. Data URIs

are capable of containing any given user input, introduced by the schema handler `data`: followed by a MIME type declaration with optional encoding parameters. An sample attack against the publicly available demonstration application of Helios is shown in Listing 1.3.

```
https://vote.heliosvoting.org/helios/nocookies?
  continue_url=data:text/html,<script>alert(1)</script>
```

Listing 1.3. A malicious link introducing an injection

The resulting markup on the Helios application is shown in Listing 1.4, a harmless link asking the administrator to proceed to finishing the election and prepare the ballot.

```
Enable cookies and then <a
href="/helios/testcookie?continue_url=data
%3Atext%2Fhtml%2C%3Cscript%3Ealert%281%29%3C%2Fscript%3E
">proceed to preparing your ballot </a>
```

Listing 1.4. The injection result reflected on Helios

This attack requires user interaction – namely a click on the now infected link. Modern browsers such as Firefox 4 or Chrome do not allow the executed JavaScript to access the election domain and read sensitive data. Older user agents such as Firefox 3 and older Safari and Opera versions nevertheless treat data URIs less carefully and allow the rendered markup access to the origin domain. This enables a fully reflected Cross-Site Scripting (XSS) attack including theft and manipulation of sensitive election data – either from the logged in voter or administrator. A different attack yields a much higher impact and resided in a bug we discovered embedded in the actual voting process, having the application attempt to use the data supplied by the parameter `election_url` to fuel the voting application with a JavaScript Object Notation (JSON) literal containing the necessary data for displaying the questions or candidates. The request URI necessary to fetch the JSON contains the election ID - a string usually carried on between administration pages via GET, thus publicly visible and easy to retrieve for an attacker. An example snippet of an infected JSON file is shown in Listing 1.5. When opening the URL directly, the name of the election will be reflected unfiltered, since the application assumes that in a JSON context no additional HTML encoding is necessary. This vulnerability could be used to conduct a full stack XSS attack allowing the attacker to obtain read and write access to the whole election process and a voters, or even administrators account. The approach worked on all tested user agents, including Internet Explorer 9, Firefox 4, Safari 5, and Opera 11.

The vulnerability enabled an attacker to craft a dummy election on the same voting server as the targeted election, send the link to his or her very own JSON file to the logged in victim and thus perform a Cross-Site Scripting attack using a persistent XSS – even avoiding detection by client side filters. In a worst case

scenario an attacker could steal sensitive information, change the displayed data or the whole ballot and election configuration and even get hands on the logged in administrators credentials by extracting cookie data and plain text password via XSS. Note that the JSON file is publicly requestable, depending on the election configuration, and capable of harming any user being logged in on the election server.

```
[{"answer_urls": [null, null, null, null], "answers":
["Yes", "No", "<img src=x onerror=alert(document.cookie)>"]
...  }]
```

Listing 1.5. An infected Helios JSON file

In general the Helios application should consider user generated output as untrusted and encode it properly – no matter in which context it is being displayed. Also each of the several parameters allowing specification of URLs to load data from or redirect to should be validated internally before being used. Otherwise open redirects, phishing attacks, and worse can result.

2.3 Extracting Sensitive Data via CSS

During our tests we noticed that in most situations critical characters, able to introduce XSS attacks, are being encoded properly. This is especially the case for characters such as *U+0022*, *U+003C* and *U+003E*. There are also couple of other special characters – namely the characters *U+007B* and *U+007D* (curly brackets). These two characters have semantic use in Cascaded Style Sheets (CSS) and mark the delimiters for a CSS property value assignment block preceded by a selector. An attacker can inject the code sequence `{ }*{font-family:{` which will, in the eyes of the user agent, turn the Helios website into a valid style-sheet that can further be included by a malicious website. When this malicious website is then visited by the targeted user or administrator, it will apply any text behind the last *U+007B* character of the injected string as the font family for any element in its context. The attacker can now use JavaScript code running in the context of the visiting user to extract the CSS `font-family` data and send it to any arbitrary domain – an example shown in Listing [1.6](#) demonstrates the data extraction involved in this attack. This enables large scale data theft from the Helios election website including sensitive data such as candidates available in the election and the user’s vote, as well as election IDs, CSRF tokens, and other values that should not leave the Helios domain context. Note that with a stolen CSRF token an attacker can easily remote administrate an election by luring the administrator to a malicious website which then will auto-submit forms with arbitrary form element values on behalf of the administrator – fully transparent and without user interaction. To fix this issue it is highly recommended to encode those *curly* characters to hexadecimal or decimal HTML entities – including the character *U+0040* capable of initiating `@charset` and `@import` directives. The attack was documented and discussed by Barth et al. and Phung [\[12,18\]](#).

```

<script>window.onload = function () {
  if (window.getComputedStyle) {
    var value = window.getComputedStyle(test, null)
      .getPropertyValue('font-family');
  } else {var value = test.currentStyle.fontFamily}
  alert(value + document.styleSheets[0].
    cssRules.item(0).cssText);
}</script>

```

Listing 1.6. Accessing CSS properties via JavaScript

2.4 Real Life Impact

Each of the aforementioned exploitation techniques already poses high risk for a Helios voting application and its users. Nevertheless we decided to craft and discuss a real life attack scenario to underline the impact of combining the discussed attacks in Section 2.1, 2.2, and 2.3. By using the available techniques an attacker is able to inject malicious code for inter alia compromising the account of a Helios voting system administrator. This can be done by luring the administrator into clicking a malicious link, which leads to loading JavaScript code into the administrators browser context to eavesdrop on every keystroke entered (JS keylogger) or extract the content entered in a specific part of the website (see Section 2.2). After exfiltrating the administrator's credentials, the authenticity of the administrator account is no longer given.

As the attacker gained control over the administrative interface, a modification of the respective election is possible. Thus, an attacker can act in the context of this user and influence wide parts of the survey assets. A XSS vulnerability found in the questions section the user is confronted with again allows for the inclusion of arbitrary code that will be executed in the voters context, as soon as he or she accessed the respective section during the voting process. By now the attacker has gained complete control over the interaction between the user and the Helios voting system, as he or she can modify the user input before it reaches the voting application and can also modify the output delivered by the voting system before the user receives it. The most simple application of this power can be referred to as a social engineering DoS (Denial-of-Service) attack: while the Helios applications continues to work underneath, the user is presented with a new layer controlled by the attacker using Clickjacking-techniques. Examples for attacks like these include showing a blank frame, non-suspicious error messages overlapping the requested web page or an overlay that informs the user that the survey is already expired or can be found at a different location.

Based on the position gained by the attacker so far, the verification specifications [19] will still work, but not necessarily on the correct data. The voter is unable to verify, if the vote was captured correctly, as the attacker controls the visual interface the user reviews and can present an output that is consistent with the user input captured before, while the input passed to the voting system can differ. Especially an unobservant voter has no reason to suspect that anyone

tampered with their vote. Because of the adversarially controlled visual interface of the Helios voting system, the voter may be duped to follow a bad link to a ballot verifier site that is controlled by the adversary. In this way there will be no hope to detect a bad ballot. On the other hand, if the voter starts a new browser session and goes independently to a ballot verifier web-site copy pasting the bad encryption data then the attacker is caught.

The possibility for an attacker to compromise a Helios administrator account yields further problems: Assuming an attacker is able to inject scripting code as shown in Section 2.2 it is possible to de-anonymize the user. This can be done by using HTML5 geo-location techniques, overlaying elements with UI redressing attacks to capture user input [1], setting cross-domain and flash cookies [20] to track a voter on websites accessed after or during the voting process, and inter alia utilize CSS-based history detection [21][22]. Allowing an attacker to gain access to the voter's browsing history and other information can thus lead to de-anonymization and is another example of threats birthing from scripting attacks against browser based voting systems.

3 Mitigation Techniques

The following section will outline several proposed protection mechanisms to help Helios-based voting applications provide better protection of the voters and their data security and integrity. The most important issue to allude is the lack of header directives enabled by default to prevent the most serious UI redressing techniques such as Clickjacking.

Recommended Headers for Helios-Based Voting Applications

- HTTP/1.1 200 OK
- Content-Type: text/html; charset=utf-8
- Transfer-Encoding: chunked
- Connection: keep-alive
- Keep-Alive: timeout=20
- Vary: Accept-Encoding
- Cache-Control: max-age=0, private, must-revalidate
- X-UA-Compatible: IE=Edge,chrome=1
- X-Frame-Options: SAMEORIGIN
- X-XSS-Protection: 1;mode=block;
- X-Content-Type-Options: nosniff

Note that not all the headers listed above are necessarily security relevant and thus might not be discussed in the following paragraphs. Most importantly the XSS protection headers have been introduced, signaling a browser to block script execution in case an injection was detected [23]. This is important - since omitting this header completely will make the application prone to attacks against an un-patched Internet Explorer 8 and a bug in its XSS filter, unveiled in 2009 [24].

The `X-Content-Type-Options` header will forbid user agents to sniff content types, in case mismatches between extension, MIME type, and file name are being spotted. This kind of attack has been seen rather often in real life and can possibly be used against the platforms' image upload tool and other components [25]. The header setting might as well help prevent XSS attacks delivered via JSON files – in case the correct MIME type is given by the server. Otherwise an attacker can abuse the content type sniffing features of common user agents to execute active HTML in arbitrary file types such as discussed in subsection 2.2. Also worth considering is the implementation of a framekiller fall-back composed in JavaScript making sure that the website cannot be framed. This is interesting for the reason that the HTTP header was introduced by Microsoft in 2008 and therefore is not supported by older web browsers like Internet Explorer 7 or Firefox 2 [26]. An academic publication by Jackson et al. sheds more light on proper ways to install JavaScript-based frame-busting code and how to avoid common pitfalls [27]. Another suggestion regarding such a frame-buster was developed by Jason Li et al. [28]. The DOM property `window.name` should be set to an unguessable random value changing with every request. This makes sure the application cannot be opened in a named frame or pop-up window and later be remotely controlled by the attacker via a different frame, tab, or website. While a header and JavaScript-based frame buster provides better protection against framing attacks, a pop-up attack can not be easily mitigated with these header directions. It has proven helpful in our tests to have a website check for the existence of a global opener variable - and deny operations in case it was found, thus indicating the presence of a potentially malicious website spawning the victim window in a pop-up.

The analysis of the demo website markup showed that in the header area two of the `<meta>` tags were separated by two newlines. This is actually a very well working protection against MHTML-based attacks, where an attacker transforms the website into a valid MHTML document, and opens it in Internet Explorer with the prefix `mhtml:` and a trailing file identifier such as `!xss.html`. These kinds of attacks have been rarely documented, information can be found on the HTML5 Security Cheatsheet [29]. It should be noted for further development iterations that every HTML view possibly containing user generated data should have two newlines in a very early location of the HTML header. We consider this protection mechanisms to be accidentally in place – since this kind of attack is not very well known. Finally the set of escaped and encoded characters should be extended to decrease and mitigate the risk of Cross-Site Scripting attacks and CSS-based information theft. This holds for critical characters in any context, as illustrated by the attack described in subsection 2.2 as well as subsection 2.3. Also the CSRF protection should be reviewed closely to avoid problems with the application ignoring faulty or even missing CSRF tokens in critical forms. We noticed that most of the forms used by Helios are applied with a CSRF token to prevent remote submission and data manipulation, but sometimes simply removing the token did surprisingly not have *any* effect on the form submit. This means an attacker can easily submit forms on behalf of a logged in user lured on

a specially prepared website. Our tests showed that some forms do not possess a protecting CSRF token at all – it is highly recommended to enforce usage of the tokens and make sure each and every non-idempotent request is protected properly. A single gap in the protection grid can allow an attacker to start with an initial injection and use it to escalate further into the system to change critical settings after some steps. Ultimately there’s a multitude of attack patterns we did not test again yet – so the platform developers should keep awareness on what happens in the web and browser security research community and harden Helios accordingly.

4 Related Work

There are many publications focusing on the attack of booth-based e-voting systems, where the paper ballot is replaced by an electronically submitted ballot. However, the voter still casts a ballot at a polling place using a dedicated computer system. For example, Bannet et al. use a simplified direct recording electronic (DRE) voting system called Hack-a-Vote to demonstrate how these systems are susceptible to manipulation [30]. Diebold’s AccuVote-TS DRE has been subject to a source code analysis by Kohno et al. [31]. They discovered that, besides other security problems, voters can cast unlimited votes without being detected and without having insider privileges. Feldman et al. analyzed a live Diebold AccuVote-TS used in US elections and point out further vulnerabilities in software and hardware that could allow for vote stealing and viral spreading of malicious code from machine to machine [32]. Gonggrijp and Hengeveld studied the Nedap/Groenendaal ES3B voting computer used in elections in the Netherlands, Germany, and France [33]. They show that brief access to the machine before an election suffices to gain complete and virtually undetectable control over the election outcome. Bishop and Wagner state that, while assessing voting systems certified for use in California, they found that the systems appeared not to be designed with security in mind [34]. Appel et al. analyse the Sequoia AVC Advantage DRE voting system and find it vulnerable to a variety of attacks, including unnoticed vote stealing [35]. Balzarotti et al. develop a methodology for attacking electronic voting systems [36] that can partially be abstracted for attacking remote or Internet voting systems.

Volkamer et al. point out [37] that Internet voting systems deal with a much less controlled environment than booth-based e-voting systems. This is underlined by the fact that the entity responsible for the election has commonly no or limited control over the client used by the voter. The authors propose a solution where trust in a certain system configuration is assured by means of trusted computing. They also point out that the common cryptographic measures of resilient voting protocols are not enough, as client-side manipulations take place before cryptographic operations are applied. Joaquim et al. [38] argue along similar lines, as the main concern of many protocols proposed so far is vote manipulation at server side, while assuming a trusted client. Burmester and Magkos [39]

as well as Pasquinucci [40] also acknowledge that even if it is possible to assure that the server the voting application is running on is configured correctly and kept up to date the same can not be assured for the voters computer and browser. The integrity of the voters computer however is a basic assumption in the design of RIES, the Rijnland Internet Election system [41]. RIES was attacked by Gonggrijp et al. [42] who found several weaknesses within the source code, among them opportunities for Cross-Site-Scripting and SQL injection due to a lack of user input validation.

In addition to RIES, prior versions of Helios have also been subject to attacks both on design and implementation level. Cortier and Smyth present an attack that violates the security objective of ballot secrecy within Helios [43]. This vulnerability will be fixed in Helios 3.1 [44]. Helios 2.0 was prone to an attack that exploits a known vulnerability in Adobe Reader to install a malicious browser extension [8]. The extension was capable of manipulating the ballot cast by the user. However, the approach exploits a vulnerability previously and explicitly acknowledged by the Helios authors [45]. In contrast, the attacks presented by us do neither rely on a flawed third-party browser extension nor the installation of malicious code on the voter's computer, but make use of features available in every recent browser.

5 Conclusion

While the back-end of Helios based voting applications might be provable secure, the front-end so far is not. Even with a decent protection level against common web application attack techniques there remain a lot of unpublished and less known ways to get hands on sensitive information and interfere with critical processes. The client side security of a system such as Helios should be considered with the same closeness as its cryptographic features and ways of submitting sensitive voter data to the voting verifier and servers.

Web application security is, considering that its actual outcome is affected by many layers, a troubled field. Whereas many high-profile websites and critical web applications ensure that network, protocol, and application layers are well secured by providing a state of the art protection, the client side is often being forgotten or treated insufficiently. Web developers have yet to fully grasp that the weakest link in the security chain is still the browser and the user operating on it.

Not only due to the mentioned reasons, securing web applications is not an effortless task – facing the unpredictable cloud of outdated, insecure, and quirky browser implementations. A voting system attackable via browser quirks, legacy features, and vulnerabilities might fuel either simple and less relevant surveys or in the worst case actual political elections. Attackers can use simple tricks to attack even highly secure web applications by tricking their users perceptions. We suggest to consider voting applications as critical components and make sure the environment, which is running the user agent, as well as the user agent itself, is trusted and can not be modified by adversaries with malicious intent.

After we spotted the aforementioned security bugs and implementation problems we contacted Helios' maintaining author B. Adida to go through the discussed bugs and consider the suggested fixes. At the time of writing most of the vulnerabilities have already been approved and addressed for in-time fixes. The Helios application should therefore soon be a more secure and trustworthy application, capable of handling the immense responsibility a browser based software voting system demands. It is to be noted that Helios being open sourced and openly testable via a public demonstration application was highly beneficiary and enabled us to spot, describe, and submit the issues we spotted; many other applications suffer from the principle security by obscurity and thus it is difficult for them to gain a similar security level as Helios.

References

1. Adida, B.: benadida/helios-server - GitHub (2011), <https://github.com/benadida/helios-server>
2. Adida, B.: Helios: Web-based open-audit voting. In: Proceedings of the 17th USENIX Security Symposium, Security 2008 (2008)
3. Mozilla Foundation: LiveConnect (MDC Documentation) (2011), <https://developer.mozilla.org/en/LiveConnect>
4. Haber, S., Benaloh, J., Halevi, S.: The Helios e-Voting Demo for the IACR (2010), <http://www.iacr.org/elections/eVoting/heliosDemo.pdf>
5. Johns, M.: Code Injection Vulnerabilities in Web Applications - Exemplified at Cross-site Scripting. PhD thesis, University of Passau, Passau (2009)
6. Balduzzi, M.: New insights into clickjacking. In: OWASP AppSec Research (2010)
7. Stone, P.: Next Generation Clickjacking (2010), <https://media.blackhat.com/bh-eu-10/presentations/Stone/BlackHat-EU-2010-Stone-Next-Generation-Clickjacking-slides.pdf>
8. Estehghari, S., Desmedt, Y.: Exploiting the client vulnerabilities in internet E-voting systems: hacking Helios 2.0 as an example. In: Proceedings of the 2010 International Conference on Electronic Voting Technology/Workshop on Trustworthy Elections, EVT/WOTE 2010 (2010)
9. Niemietz, M.: UI redressing: Attacks and countermeasures revisited (2011), <http://ui-redressing.mniemietz.de/uiRedressing.pdf>
10. Raskin, A.: Tabnabbing: A new type of phishing attack (2010), <http://www.azarask.in/blog/post/a-new-type-of-phishing-attack/>
11. Krebs, B.: Devious new phishing tactic targets tabs (2010), <http://krebsonsecurity.com/2010/05/devious-new-phishing-tactic-targets-tabs/>
12. Barth, A., Caballero, J., Song, D.: Secure content sniffing for web browsers, or how to stop papers from reviewing themselves. In: Proc. of the 30th IEEE Symposium on Security and Privacy (Oakland 2009), Oakland (2009)
13. Stamm, S., Sterne, B., Markham, G.: Reining in the web with content security policy. In: Proceedings of the 19th International Conference on World Wide Web (2010)
14. Heiderich, M.: <malicious></markup>: HTML form controls reviewed (2008), <http://maliciousmarkup.blogspot.com/2008/11/html-form-controls-reviewed.html>

15. Phung, P.H., Sands, D., Chudnov, A.: Lightweight Self-Protecting javascript. In: ACM Symposium on Information, Computer and Communications Security (ASI-ACCS) (March 2009)
16. OWASP: Enterprise security API (2011), http://www.owasp.org/index.php/Category:OWASP_Enterprise_Security_API
17. Masinter, L.: RFC 2397 - the "data" URL scheme (1998), <http://www.ietf.org/rfc/rfc2397.txt>
18. Huang, L., Weinberg, Z., Evans, C., Jackson, C.: Protecting browsers from Cross-Origin CSS attacks. In: Proc. of the 17th ACM Conference on Computer and Communications Security, CCS 2010 (2010)
19. heliosvoting.org: Helios v1 and v2 Verification Specs (2011), <http://documentation.heliosvoting.org/verification-specs/helios-v1-and-v2-verification-specs>
20. Ayenson, M., Wambach, D.J., Soltani, A., Good, N., Hoofnagle, C.J.: Flash cookies and privacy ii: Now with html5 and etag respawning (2011), http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1898390
21. Janc, A., Olejnik, L.: Web Browser History Detection as a Real-World Privacy Threat. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) ESORICS 2010. LNCS, vol. 6345, pp. 215–231. Springer, Heidelberg (2010)
22. Weinberg, Z., Chen, E.Y., Jayaraman, P.R., Jackson, C.: I still know what you visited last summer (2011), <http://websec.sv.cmu.edu/visited/visited.pdf>
23. Ross, D.: IE8 security part IV: the XSS filter - IEBlog (2008), <http://blogs.msdn.com/b/ie/archive/2008/07/02/ie8-security-part-iv-the-xss-filter.aspx>
24. Maone, G.: IE's XSS filter creates XSS vulnerabilities (2009), <http://hackademix.net/2009/11/21/ies-xss-filter-creates-xss-vulnerabilities/>
25. MSDN: MIME type detection in internet explorer (2011), [http://msdn.microsoft.com/en-us/library/ms775147\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/ms775147(v=vs.85).aspx)
26. Mozilla Foundation: The X-Frame-Options response header (MDC Documentation) (2010), https://developer.mozilla.org/en/the_x-frame_options_response_header
27. Rydstedt, G., Bursztein, E., Boneh, D., Jackson, C.: Busting frame busting: a study of clickjacking vulnerabilities on popular sites. In: Web 2.0 Security and Privacy 2010 (W2SP 2010) (2010)
28. Li, J., Schmidt, C., Crawford, B.: Clickjacking defense (2011), <https://www.codemagi.com/blog/post/194>
29. Silin, A.: HTML5 security cheatsheet: MHTML Attacks (2011), <http://html5sec.org/?mhtml>
30. Bannet, J., Price, D.W., Rudys, A., Singer, J., Wallach, D.S.: Hack-a-vote: Security issues with electronic voting systems. IEEE Security & Privacy 2, 32–37 (2004)
31. Kohno, T., Stubblefield, A., Rubin, A.D., Wallach, D.S.: Analysis of an electronic voting system. In: Proceedings of the 25th IEEE Symposium on Security and Privacy, Oakland 2004 (2004)
32. Feldman, A.J., Halderman, J.A., Felten, E.W.: Security analysis of the Diebold AccuVote-TS voting machine. In: Proceedings of the USENIX Workshop on Accurate Electronic Voting Technology (2007)
33. Gonggrijp, R., Hengeveld, W.: Studying the Nedap/Groenendaal ES3B voting computer: a computer security perspective. In: Proceedings of the USENIX Workshop on Accurate Electronic Voting Technology (2007)
34. Bishop, M., Wagner, D.: Risks of e-voting. Communications of the ACM 50 (2007)

35. Appel, A.W., Ginsburg, M., Hursti, H., Kernighan, B.W., Richards, C.D., Tan, G., Venetis, P.: The new jersey voting-machine lawsuit and the AVC advantage DRE voting machine. In: Proceedings of the 2009 Conference on Electronic Voting Technology/Workshop on Trustworthy Elections, EVT/WOTE 2009. USENIX Association (2009)
36. Balzarotti, D., Banks, G., Cova, M., Felmetzger, V., Kemmerer, R., Robertson, W., Valeur, F., Vigna, G.: An Experience in Testing the Security of Real-World Electronic Voting Systems. *IEEE Transactions on Software Engineering* 36 (2010)
37. Volkamer, M., Alkassar, A., Sadeghi, A.R., Schulz, S.: Enabling the application of open systems like PCs for online voting. In: Proceedings of the Workshop on Frontiers in Electronic Elections 2006 (2006)
38. Joaquim, R., Ribeiro, C., Ferreira, P.: Improving Remote Voting Security with CodeVoting. In: Chaum, D., Jakobsson, M., Rivest, R.L., Ryan, P.Y.A., Benaloh, J., Kutylowski, M., Adida, B. (eds.) *Towards Trustworthy Elections*. LNCS, vol. 6000, pp. 310–329. Springer, Heidelberg (2010)
39. Burmester, M., Magkos, E.: Towards secure and practical E-Elections in the new era. In: *Secure Electronic Voting*. *Advances in Information Security*, pp. 63–76 (2003)
40. Pasquinnucci, A.: Web voting, security and cryptography. *Computer Fraud & Security* 2007, 5–8 (2007)
41. Hubbers, E., Jacobs, B., Schoenmakers, B., van Tilborg, H., de Weger, B.: Description and analysis of RIES (2008), http://www.win.tue.nl/eipsi/images/RIES_descr_anal_v1.0_June_24.pdf
42. Gonggrijp, R., Hengeveld, W.-J., Hotting, E., Schmidt, S., Weidemann, F.: RIES - Rijnland Internet Election System: A Cursory Study of Published Source Code. In: Ryan, P.Y.A., Schoenmakers, B. (eds.) *VOTE-ID 2009*. LNCS, vol. 5767, pp. 157–171. Springer, Heidelberg (2009)
43. Cortier, V., Smyth, B.: Attacking and fixing Helios: An analysis of ballot secrecy. *Technical Report 2010/625* (2010)
44. Adida, B.: *Attacks and Defenses - Helios* (2011), <http://documentation.heliosvoting.org/attacks-and-defenses>
45. Adida, B., De Marneffe, O., Pereira, O., Quisquater, J.: Electing a university president using open-audit voting: analysis of real-world use of helios. In: Proceedings of the 2009 Conference on Electronic Voting Technology/Workshop on Trustworthy Elections (2009)

An Efficient and Highly Sound Voter Verification Technique and Its Implementation

Rui Joaquim¹ and Carlos Ribeiro²

¹ Inesc-ID \ISEL

Rua Conselheiro Emídio Navarro 1, 1959-007 Lisboa, Portugal

`rjoaquim@cc.isel.pt`

² Inesc-ID \UTL

Rua Alves Redol 9 - 6 andar, 1000-029 Lisboa, Portugal

`carlos.ribeiro@ist.utl.pt`

Abstract. This paper presents MarkPledge3 (MP3), the most efficient specification of the MarkPledge (MP) technique. The MP technique allows the voter to verify that her vote is correctly encrypted with a soundness of $1 - 2^{-\alpha}$, with $20 \leq \alpha \leq 30$, just by performing a match of a small string (4-5 characters). Due to its simplicity, verifying the election public data (vote encryptions and tally) in MP3 is 2.6 times faster than with MP2 and the vote encryption creation on devices with low computational power, e.g. smart cards, is approximately 6 times better than the best of the previous MP specifications (MP1 and MP2).

Keywords: Verifiability, Voter Vote verification, MarkPledge.

1 Introduction

The MarkPledge (MP) technique was introduced by Neff in 2004 [22] with the goal of providing high vote encryption assurance to the voter, i.e. give the voter high certainty that the encrypted vote, generated by the voting machine, is an encryption of the voter's choice. In its essence MP defines how to encrypt two types of votes: a vote in favor of a candidate, a *YESvote*, and a vote against/neutral to a candidate, a *NOvote*. The MP candidate vote encryption is special because it contains random data that is used to create a verification code, which can to prove to the voter the type of the candidate vote encryption. The voter verifies that a candidate vote encryption is in fact a *YESvote* by doing a short string match. The verification of a *NOvote* usually requires some extra effort from the voter, but can be made unneeded by the specific vote protocol where it is used.

In MP based vote protocols [1,3,4,19,22], the voter's choice is encrypted with a *YESvote*, for the selected candidate, and with several independent *NOvotes* for the non selected candidates. Then, a vote receipt is created with the verification codes of all candidate vote encryptions. To simplify the voter's receipt verification, the vote protocol provides a mathematical proof that there is only one *YESvote* in the set of candidate encryptions, therefore the voter only needs to verify the *YESvote* candidate encryption.

The soundness of the voter verification process is $1 - 2^{-\alpha}$, where α is a configurable security parameter that defines the size in bits of the verification code to match. To achieve a usability *vs* security balance, α is usually set to a value between 20 and 30, corresponding to a verification code of 4 to 5 characters.

The high soundness of the voter’s receipt verification is only guaranteed if the vote encryption is valid and the vote receipt correct, i.e. if there is only one *YES* vote and if all verification codes match the corresponding individual candidate vote encryptions. However, the proofs of vote validity and receipt correctness require some complex math, which the common voter cannot perform. Thus, the MP technique, and the vote protocols that use it, define public verifiable vote validity and receipt correctness proofs to protect the voter’s privacy. Anyone with the sufficient knowledge and computational power can verify the validity and correctness of all vote-receipt pairs.

Our major contribution is a faster MP solution (MP3) that can be proven to be as sound and privacy-keeping as any of previous MP solutions [4,22], without consuming more memory. Both previous MP solutions [4,22] have high computational vote generation costs, which makes them unsuitable to be used in mobile voting scenarios where the voting machine has low computational power, e.g. a smart-card or a secure element of a mobile phone, both usually standard JavaCards. MP3 also offers a considerable 2.6 times improvement on the public vote-receipt validity and correctness verifications over the best previous solution (MP2). This improvement enlarges the number of public organizations with enough computer power to verify all the votes of a national general election .

Our second contribution is an abstraction layer for the MP technique, composed of 5 functions: the vote encryption function \mathcal{VE}_{pk} , which creates the candidate vote encryption; the vote receipt creation function \mathcal{RC}_{pk} , which given a candidate encryption generates the corresponding verification code; the vote validity function \mathcal{VV}_{pk} , which verifies the validity of a candidate encryption; the receipt validity function \mathcal{RV}_{pk} , which validates the correspondence between a candidate vote encryption and a verification code; and, finally, a canonicalization function \mathcal{C}_{pk} which prepares the candidate vote encryption for the vote tally process. The MP abstraction layer adds nothing to the MP solutions (MP1, MP2 and MP3) or to the MP based vote protocols. It only identifies common processes to all MP solutions, thus, it facilitates the comparison of the different MP solutions and their substitution in a MP based vote protocol.

We have partially implemented each one of the three MP solutions (MP1, MP2 and MP3) in two types of smart-cards, a MULTOS smart card and a JavaCard. The former is faster, but the latter is more ubiquitous, being deployed in secure elements of recent mobile phones and many National Identity Cards. In both cases MP3 is the only viable solution given that the time required to vote with MP1 and MP2 exceeds the time a user will be, usually, willing to wait.

The next section presents the related work and describes the simplified version of a MP vote protocol. Sections 3 and 4 describe the new MP3 proposal and present a detailed description of its cryptographic functions. Section 5 provide a comparative analysis of all MP solutions. Finally, we conclude in section 6.

2 Related Work

The research on electronic voting protocols always had the goal to provide verifiable protocols [7,8,13,14,16,17,20,24]. However, the verification procedures are usually too difficult for a human to do, requiring the use of a trusted Vote Machine (*VM*) to help the voter in the process. In 2004, Chaum [9] and Neff [22] (MarkPledge) introduced techniques that enable a human to verify a cryptographic vote, eliminating the need for a trusted *VM*.

In the original Chaum's work [9] the voter verifies her vote through a two-sheet ballot print, by a special printer, that uses transparent sheets and visual cryptography to show the voter choices in a human readable format. The voter then destroys one sheet and keeps the other as a verifiable privacy-preserving receipt. This procedure models a simple "cut-and-choose" technique giving the voter a $\frac{1}{2}$ probability to detect a fraud. Punchscan [10] and Pret a Voter [12] simplify the Chaum's system setup by using a simpler pre-printed ballot. In both systems the voter still have only a probability of $\frac{1}{2}$ to detect any fraud with her vote. Both systems allow for pre-election cut-and-choose verification aiming to reduce the danger of a large scale fraud. The verification procedure consists in printing an excess of ballots and then randomly auditing the extra ballots.

Adida and Neff [3] and Joaquim et al. [19] present simplifications to the voter interaction of MP, improving its usability. In 2006, and also based on the MP construction, Moran and Naor presented a voter verifiable voting system with everlasting privacy [21], replacing the vote encryption with vote commitments. The main disadvantage of the original MP specification (MP1) is the high computational costs of the technique, specially when compared to a vote protocol that encrypts the vote as a simple encryption of the candidate identifier. In MP1 the vote is encrypted with $2 \cdot \alpha$ encryptions for each candidate. A direct consequence of the MP1 vote encryption structure is that the vote encryption needs $2 \cdot \alpha \cdot k$ more disk space when compared to a simple vote encryption, where k is the number of running candidates. The performance issues of MP were first addressed in MP2 [4]. However, MP2 is still too heavy for low computational power devices (cf. section 5.1).

A completely different voter verification approach was proposed by Benaloh in [5,6]. His proposal consists in separating the vote encryption process from the vote casting process. The *VM* is responsible only for the vote encryption, which it delivers to the voter (e.g. in a paper receipt). The voter can then choose to cast the vote or to verify it by asking to decrypt the encrypted vote. In this solution the voter can verify as many vote encryptions as she wants until she gains confidence in the *VM*. In theory, this approach can have the same $1 - 2^{-\alpha}$ MP soundness if a voter is allowed to use the *VM* to create 2^α independent vote encryptions. This procedure is clearly unpractical and would yield a computational cost much higher than the one of the original MP. The ideas of Benaloh's work were used for the voter verification mechanisms used by the VoteBox [25] and Helios [2] voting systems.

2.1 MarkPledge Simplified Vote Protocol Overview

Usually, a voter has no way to be assured that the cryptographical representation of her vote, produced by the voting machine, encodes her candidate choice. MP attains that goal by performing a zero-knowledge proof (ZKP) with the voter herself, not with some proxy, that the encrypted vote for the chosen candidate is, in fact, a *YESvote*. In order to ensure receipt-freeness a simulated ZKP is conducted for every other candidate encryption (*NOvotes*), in such a way that only the voter is able to tell which is the real proof among the simulated ones.

More precisely, zero-knowledge, in the MP context, means that it is not possible to identify the type of a candidate encryption from the corresponding public data: candidate encryption, verification code and corresponding mathematical proofs of vote validity and verification code correctness. To protect the voter's privacy, the correctness verification is the same for every candidate encryption, whether it bares a *YESvote* or a *NOvote*. In fact, the correctness verification is, in both cases, a ZKP that aims to prove that the candidate encryption bares a *YESvote*. The proof is only real if the candidate encryption bares a *YESvote*. This is because only the commit value (verification code) of the ZKP of the *YESvote* is shown (pledged) to the voter. The commit values (verification codes) of the ZKP of the *NOvotes* are not shown to anyone a may therefore be crafted so that the ZKP on *NOvotes* seem real although they are simulated.

To clarify the MP technique use within a vote protocol follows a short description of the simplified vote protocol of [3]. Note however, that each MP version (MP1, MP2 and MP3) may be used in different voting protocols, e.g. the original one [22], the simplified versions [3,22], and the Internet protocol of [19]. The described MP vote protocol has four phases [3]: the first three match the usually ZKP (commit, challenge, validation), and the last one is an anonymization and counting step. The following protocol description also introduces the MP abstraction layer, i.e. it shows where each of the five MP functions are used. The specification of each function is given in section 4, for the MP3 solution, and in an extended version of this paper [18], for the MP1 and MP2 solutions.

Phase 1. Vote Encryption

1. The vote machine (*VM*) presents the list of candidates to the voter.
2. The voter enters her vote selection (candidate j).
3. The *VM*, using the candidate vote encryption function $\mathcal{V}\mathcal{E}_{pk}$, creates the vote encryption as a sequence of individual MP candidate vote encryptions (dubbed as bit encryptions, $BitEnc(b)$ in [1,3,4,19]). The selected candidate (candidate j) gets a *YESvote* = $BitEnc(1)$, and each other candidate gets an individual *NOvote* = $BitEnc(0)$.

Each candidate vote encryption $BitEnc(b)_i$ encrypts a random commit code θ_i , which later allows the voter to verify the vote encryption by matching the verification code ϑ_j in the vote receipt. At this point in the protocol only the verification code of the *YESvote* is known $\vartheta_j = \theta_j$.

4. The *VM* commits to the vote encryption, e.g. by printing or publishing it in a public bulletin board.

5. The *VM* pledges (reveals on an untappable channel) to the voter the *YESvote* verification code $\vartheta_j = \theta_j$ as the *pledge* value.

Phase 2. Receipt Creation and Voter Verification

1. The voter sends a random vote challenge c to the *VM*. Originally the challenge value was chosen by the voter herself [22]; subsequent versions remove this task from the voter with the help of a trusted third party [3, 19]. This step is crucial for the voter's verification soundness because the random challenge is what prevents the *VM* to pledge a valid verification code for a *NOvote* in the previous protocol step.
2. The *VM*, using the candidate receipt creation function \mathcal{RC}_{pk} , computes the verification codes for the non-selected candidates ($\vartheta_i : i \neq j$), i.e. the verification codes for all the *NOvotes*. Each computed verification code $\vartheta_i : i \neq j$ is the result of a function between the random commit code θ_i inside the corresponding *NOvote_i* and the challenge value c .
3. The *VM* prints/publishes in a public bulletin board the receipt as the sequence of all the verification codes ϑ_i , in the same order of the candidate vote encryptions in the vote encryption. Along with the vote receipt, the *VM* publishes the data necessary to verify the vote validity (*voteValidity_i*) and the receipt correctness (ω_i). The *voteValidity_i* and ω_i are, respectively, outputs of the \mathcal{VE}_{pk} and \mathcal{RC}_{pk} functions.
4. The voter verifies the correction of the vote encryption by verifying if the *pledge* value, pledged in step 5 of phase 1, matches the verification code ϑ_j associated to her chosen candidate (candidate j) in the vote receipt.

Phase 3. Third Party Vote/Receipt Validation

To certify the voter receipt verification, one or several third parties validate the vote/receipt pair validity and correctness, using the vote validity (\mathcal{VV}_{pk}) and receipt validity (\mathcal{RV}_{pk}) functions. The \mathcal{VV}_{pk} function attests that each *BitEnc(b)_i* is valid, i.e. that it is either a *BitEnc(0)* or a *BitEnc(1)*. The \mathcal{RV}_{pk} function attests that each verification code ϑ_i corresponds to *BitEnc(b)_i*. Both certifications are performed only on public data, thus they do not compromise the voter's privacy. Optionally, using the techniques described in section 4.1, the third parties can also verify that the encrypted vote has only one *YESvote*. Without this verification the *VM* can create invalid votes, i.e. votes with more than one *YESvote*. Although, several MP based vote protocols omit this verification step and only verify that there is only one *YESvote* in the vote encryption in the tally phase, after the anonymous vote decryption.

Phase 4. Vote Canonicalization and Counting

Finally, the vote encryptions are made uniform, by the vote canonicalization function \mathcal{C}_{pk} , and then anonymized and counted. The results are published in a public bulletin board.

The canonicalization process is necessary because every candidate vote encryption encrypts a random one-time commit code (θ_i), that is used to deterministically compute the verification codes of the *BitEnc*(b)s. Thus, revealing the θ_i commit codes would enable to identify the voter’s candidate choice by a simple correlation with the verification codes ϑ_i in the voter’s vote receipt.

The vote canonicalization process is public and therefore verifiable. Once all vote encryptions are in the canonical form it is possible to decrypt them and compute the election vote tally, usually using a mix-net or a homomorphic vote tally process to protect the voter’s privacy.

The *BitEnc*(b), *voteValidity* and ω details, and the procedures to verify the vote validity and receipt correctness, are presented in sections 3 and 4, for the MP3 solution, and in the extended version of this paper [18] for the MP1 and MP2 solutions.

3 MarkPledge3

The key aspect of every MP system is the two step verification of the vote encryption, carried by the *VM* to the voter, to prove that the encrypted vote expresses the voter’s intentions, cf. phases 2 and 3 of the protocol described in section 2.1. This section describes the key insight of this two step proof verification in MP3. It starts by describing the cryptosystem used by MP3, and continues by showing how its homomorphic properties are used to implement the candidate vote encryption, the proof and the two step proof validation.

3.1 Homomorphic Cryptosystem Details

The MP3 implementation is based on the homomorphic properties of the ElGamal cryptosystem [15]. The ElGamal cryptosystem works in the \mathbb{Z}_p^* subgroup G_q of order q , where p and q are large primes such that $q|p-1$. Both primes p , q and a generator g of G_q are public parameters of the system. The ElGamal key pair consist of a private key s and the corresponding public key $h = pk = g^s \bmod p$. The private key s is a randomly chosen integer such that $0 < s < q$. Algorithms to generate secure ElGamal parameters can be found in [23].

A message $m \in G_q$ is encrypted by selecting a random integer value $r \in \mathbb{Z}_q$, and constructing the following pair $\mathcal{E}_{\mathcal{G}_h}(m, r) = \langle x, y \rangle = \langle g^r \bmod p, h^r \cdot m \bmod p \rangle$. To recover the message m one computes $m = \frac{y}{x^s}$. In order to have the desired homomorphic properties we use a variant known as exponential ElGamal [14]. In exponential ElGamal the message to encrypt m is chosen from \mathbb{Z}_q and it is encrypted as g^m , instead of m , in order to respect the ElGamal message space, i.e. $\mathcal{E}_h(m, r) = \mathcal{E}_{\mathcal{G}_h}(g^m, r)$. The exponential ElGamal has the following homomorphisms (we have omitted the *mod p* notation from the equations):

Additive homomorphism between two encryptions

$$\begin{aligned} \mathcal{E}_h(m_1, r_1) \oplus \mathcal{E}_h(m_2, r_2) &= \langle g^{r_1}, h^{r_1} \cdot g^{m_1} \rangle \cdot \langle g^{r_2}, h^{r_2} \cdot g^{m_2} \rangle = \\ &\langle g^{r_1} \cdot g^{r_2}, h^{r_1} \cdot g^{m_1} \cdot h^{r_2} \cdot g^{m_2} \rangle = \langle g^{r_1+r_2}, h^{r_1+r_2} \cdot g^{m_1+m_2} \rangle = \\ &\mathcal{E}_h((m_1 + m_2) \bmod q, (r_1 + r_2) \bmod q) \end{aligned}$$

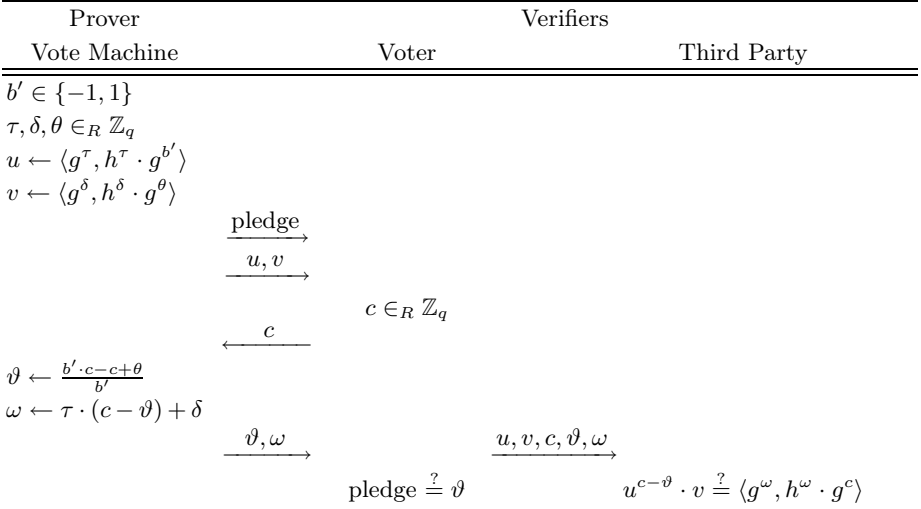


Fig. 1. MP3 *YESvote* ($b' = 1$) candidate encryption and verification protocol. The *NOvote* ($b' = -1$) case is identical, the only difference is that the voter verifies that $\text{pledge} = 2 \cdot c - \vartheta$ instead of $\text{pledge} = \vartheta$, cf. section 3.2. ϑ and ω are computed *mod q*.

Multiplicative Homomorphism between an Encryption and a Value n

$$\mathcal{E}_h(m, r) \otimes n = \langle g^r, h^r \cdot g^m \rangle^n = \langle (g^r)^n, (h^r)^n \cdot (g^m)^n \rangle = \langle g^{r \cdot n}, h^{r \cdot n} \cdot g^{m \cdot n} \rangle = \mathcal{E}_h((m \cdot n) \bmod q, (r \cdot n) \bmod q)$$

3.2 MarkPledge3 Insights

Figure 1 depicts the MP3 candidate vote encryption and verification. A MP3 candidate vote encryption $\text{BitEnc}(b) = \langle u, v \rangle$ is composed by two independent encryptions: u is the encryption of either $b' = 1$ for a *YESvote* or $b' = -1$ for a *NOvote*; v is the encryption of a random commit code (θ), which, in the case of a *YESvote*, is pledged over an untappable channel to the voter as being the corresponding verification code.

After the encrypted vote creation by the vote machine, the voter selects a random challenge c that will be used by the vote machine to create a public verifiable receipt (ϑ, ω) . ϑ is the public verification code and ω the public data that allows to prove that ϑ is the correct verification code for the pair (u, v) . ϑ correctness is guaranteed by an independent third party verification of the following equality: $u^{c-\vartheta} \cdot v = \langle g^\omega, h^\omega \cdot g^c \rangle = \mathcal{E}_{pk}(c, \omega)$. Thus, assuming that u encrypts a value $b' \in \{-1, 1\}$, which can be proved by known techniques, cf. section 4, the preceding equality and the homomorphic properties of the ElGamal cryptosystem guarantee that $\vartheta \in \{\theta, 2 \cdot c - \theta\}$, cf. equation 1. The encryption factor ω can be easily computed by the vote machine as $\omega = \tau \cdot (c - \vartheta) + \delta$.

$$\begin{aligned}
u^{c-\vartheta} \cdot v &= \mathcal{E}_{pk}(c, \omega) \Rightarrow \\
b' \cdot (c - \vartheta) + \theta &= c \Leftrightarrow \vartheta = \frac{b' \cdot c - c + \theta}{b'}
\end{aligned} \tag{1}$$

thus:

$$b' = 1 \Rightarrow \vartheta = \theta \quad \wedge \quad b' = -1 \Rightarrow \vartheta = 2 \cdot c - \theta$$

By equation [1](#), before knowing the challenge value c , the vote machine can only pledge the verification code ϑ for a *YESvote* ($b' = 1$), which is the random commit code θ encrypted in v . Thus, the voter receipt verification for a *YESvote* is just the verification that pledge = ϑ , which is a simple string match.

A *NOvote* verification requires some extra work for the voter. In this case, the vote machine pledges the value θ and the voter must reconstruct the verification code $\vartheta = 2 \cdot c - \theta$. Although, the MarkPledge based vote protocols usually do not require the voter to perform this verification by providing a proof that there is only one *YESvote* candidate encryption in the vote, cf. sections [1](#) and [2.1](#). The details on how to create such proof for MP3 are presented in section [4.1](#).

The voter verification, as described in this section, requires the match of a long string by the voter, i.e. both the pledge value and the verification code domains are \mathbb{Z}_q . This usability issue is addressed in section [4.2](#).

Finally, note that in the special case where $c = \theta$ nothing is proved about the value of b' . However, assuming that the c value is selected randomly from a large domain this possibility is negligible.

3.3 MarkPledge3 Soundness and Zero-Knowledge Properties

This section provides proof sketches for the MP3 soundness and zero-knowledge properties. The proof sketches follow the MP3 candidate encryption and verification protocol described in figure [1](#).

Theorem 1. *Under the semantic security of the ElGamal cryptosystem, MP3 offers a verification soundness of $1 - 2/q$, provided that the value u is a valid encryption, i.e. u encrypts either $b' = 1$ or $b' = -1$.*

Proof Sketch. In order to prove the soundness of MP3, we will show that the probability that the voting machine or any other adversary have to cheat the voter in believing that she has issued a *YESvote* while she has actually issued a *NOvote* is $1 - 2/q$. The converse probability of cheating the voter in believing that she had issued a *NOvote* while she had issued a *YESvote* may be easily shown to also be $1 - 2/q$.

Assuming a valid u , the validation of the equality $u^{c-\vartheta} \cdot v = \langle g^\omega, h^\omega \cdot g^c \rangle$ ensures by equation [1](#) that $\vartheta = \theta$ if $b' = 1$ (*YESvote*), and that $\vartheta = 2 \cdot c - \theta$ if $b' = -1$ (*NOvote*). Given that the voter checks that the pledge of her's vote is pledge = ϑ , the only way the vote machine or another adversary have to fool the voter is to pledge to the voter a value pledge = $\vartheta = 2 \cdot c - \theta$, without knowing the challenge c (the challenge is revealed only after the pledge). Since $c \in_{\mathcal{R}} \mathbb{Z}_q$

it is trivial to note that $\text{pledge} = 2 \cdot c - \theta$ would also have a random distribution in \mathbb{Z}_q . Therefore, the vote machine has only a probability of $1/q$ to guess such value. Finally, removing the case where nothing is proved about b' , when $c = \theta$, we can conclude that MP3 offers a vote verification soundness of $1 - 2/q$.

Theorem 2. *Under the semantic security of ElGamal cryptosystem, the candidate vote encryption verification is zero-knowledge to every one not knowing the pledge, provided that the challenge c is randomly and independently chosen from the pledge value.*

Proof Sketch. Assuming the semantic security of ElGamal cryptosystem neither u or v reveal anything about the vote encrypted in u . Identically, under the challenge c independence generation assumption, c reveals nothing about the value encrypted in u , or v . Given that $\omega = ((c - \vartheta) \cdot \tau + \delta)$ is a linear combination of the two ElGamal random factors, used to generate u and v , it can reveal one of the encryptions u or v but only if the other one is broken. Since no individual public factor used by the verification code correctness proof reveals the vote, the only other solution is testing all together with the verification code validation equation: $u^{c-\vartheta} \cdot v = \langle g^\omega, h^\omega \cdot g^c \rangle$. However, by equation [11](#), for the same combination of public values u, v, c, ϑ and ω there is always two possible scenarios for which the equation holds, i.e. the scenario in which u encrypts $b' = -1$ (*NOvote*) and v is the encryption of $\theta = 2 \cdot c - \vartheta$ is indistinguishable from the scenario in which u encrypts $b' = 1$ (*YESvote*) and v encrypts $\theta = \vartheta$.

4 MarkPledge 3 Functions Details

In order to better compare MP3 with MP1 and MP2, this section presents the MP abstraction layer, along with its MP3 specification. The MP abstraction layer is composed by a set of five functions that all MP solutions have, although these functions in the previous MP specifications were implicit in the vote protocols. Due to space constrains the functions implementations for MP1 and MP2 are presented in an extended version of this paper [18](#).

The first two functions execute the candidate vote encryption (\mathcal{VE}_{pk}) and the receipt (verification code) creation (\mathcal{RC}_{pk}). The first one corresponds to the *BitEnc*(b) encryption in MP1 and MP2, whilst the second one bares no name in MP1 and MP2 previous descriptions. The next two functions are the candidate vote (\mathcal{VV}_{pk}) and receipt (\mathcal{RV}_{pk}) validation functions. Finally, the last function (\mathcal{C}_{pk}) prepares the candidate votes for an anonymous vote tally process with a canonization process. All functions use the election's public key pk , which is usually generated by a set of trustees using a threshold scheme.

Additionally, in section [4.1](#) it is described how to verify that a set of candidate votes contains only one *YESvote* and how to perform a verifiable homomorphic vote tally with MP3. Finally, section [4.2](#) shows how to adjust the ϑ verification code to a size that is easily usable by humans.

Vote Encryption

$$\mathcal{V}\mathcal{E}_{pk}(b, \theta, r) = \langle \text{BitEnc}(b), \text{voteValidity} \rangle$$

Where:

$$\text{BitEnc}(b) = \langle u, v \rangle = \langle \mathcal{E}_{pk}(b', \tau) = g^\tau, h^\tau \cdot g^{b'}, \mathcal{E}_{pk}(\theta, \delta) \rangle$$

$$b' = \begin{cases} 1 & \text{if } b = 1 \text{ (YESvote)} \\ -1 & \text{if } b = 0 \text{ (NOvote)} \end{cases}$$

$$r = \tau \parallel \delta, h = \text{election public key (pk)}$$

$$\text{voteValidity} = \langle a_1, a_2, b_1, b_2, d_1, d_2, r_1, r_2, c \rangle$$

$$\text{if } b = 1 \text{ (YESvote)} \begin{cases} \sigma, r_1, d_1 \in_{\mathcal{R}} \mathbb{Z}_q \\ a_1 = g^{r_1 + \tau \cdot d_1}, a_2 = g^\sigma \\ b_1 = h^{r_1 + \tau \cdot d_1} \cdot g^{2 \cdot b' \cdot d_1}, b_2 = h^\sigma \\ d_2 = c - d_1, r_2 = \sigma - \tau \cdot d_2 \end{cases}$$

$$\text{if } b = 0 \text{ (NOvote)} \begin{cases} \sigma, r_2, d_2 \in_{\mathcal{R}} \mathbb{Z}_q \\ a_1 = g^\sigma, a_2 = g^{r_2 + \tau \cdot d_2} \\ b_1 = h^\sigma, b_2 = h^{r_2 + \tau \cdot d_2} \cdot g^{2 \cdot b' \cdot d_2} \\ d_1 = c - d_2, r_1 = \sigma - \tau \cdot d_1 \end{cases}$$

$$c = \text{Hash}(u, a_1, a_2, b_1, b_2)$$

The vote encryption function $\mathcal{V}\mathcal{E}_{pk}$ is used in the first phase of a vote protocol to generate each candidate vote encryption, cf. step 3 of the simplified MP vote protocol presented in section 2.1. In MP3 each candidate vote $\text{BitEnc}(b)$ is a simple pair of exponential ElGamal encryptions, dubbed u and v . The first one encrypts a value $b' \in \{-1, 1\}$ accordingly to the value of b and the second is just the encryption of $\theta \in_{\mathcal{R}} \mathbb{Z}_q$. Both encryptions use exponential ElGamal with the randomization factors τ and δ derived from the input value $r = \tau \parallel \delta$. The voteValidity data proves that u is an ElGamal exponential encryption of a value $b' \in \{-1, 1\}$. In MP3 it consists in the output of the ballot validity proof protocol of Cramer et al. [14]. In its original context, the Cramer et al. protocol proves that a vote is an ElGamal exponential encryption of a message $m \in \{-1, 1\}$, which is exactly our definition of a valid u .

Receipt Creation

$$\mathcal{R}\mathcal{C}_{pk}(\text{BitEnc}(b), r, c) = \langle \vartheta, \omega \rangle$$

Where:

$$\text{BitEnc}(b) = \langle \mathcal{E}_{pk}(b', \tau), \mathcal{E}_{pk}(\theta, \delta) \rangle$$

$$r = \tau \parallel \delta, c \in_{\mathcal{R}} \mathbb{Z}_q$$

$$\vartheta = \begin{cases} \theta & \text{if } b' = 1 \text{ (YESvote)} \\ 2 \cdot c - \theta \text{ mod } q & \text{if } b' = -1 \text{ (NOvote)} \end{cases}$$

$$\omega = \tau \cdot (c - \vartheta) + \delta \text{ mod } q$$

In the simplified protocol of section 2.1, the \mathcal{RC}_{pk} function is used in step 2 of the protocol's second phase, after the pledge has been shown to the voter and after the challenge c disclosure to the vote machine. This function generates a receipt (verification code ϑ) as explained in section 3.2, i.e. it outputs the random commit code θ , if the candidate vote is a $BitEnc(1)$ (i.e. a *YESvote*), or outputs the θ symmetric value, taking c as the symmetry axis, if the candidate vote is a $BitEnc(0)$ (i.e. a *NOvote*). The ω data is the combination of the randomization factors used in the u and v encryptions, which is needed to verify that the verification equation results in an encryption of the challenge value c , as described by the \mathcal{RV}_{pk} function below. In order to work in accordance with the ElGamal homomorphic properties, both the ϑ and ω values are computed *mod* q .

Vote Validity

$$\mathcal{VV}_{pk}(BitEnc(b), voteValidity) = validity$$

Where:

$$BitEnc(b) = \langle u = \langle x, y \rangle, v \rangle, \quad voteValidity = \langle a_1, a_2, b_1, b_2, d_1, d_2, r_1, r_2, c \rangle$$

$$validity = \begin{cases} True & \text{if } c = Hash(u, a_1, a_2, b_1, b_2) \\ & \wedge c = d_1 + d_2 \\ & \wedge a_1 = g^{r_1} \cdot x^{d_1} \\ & \wedge a_2 = g^{r_2} \cdot x^{d_2} \\ & \wedge b_1 = h^{r_1} \cdot (y \cdot g)^{d_1} \\ & \wedge b_2 = h^{r_2} \cdot (y \cdot g^{-1})^{d_2} \\ False & \text{otherwise} \end{cases}$$

$$h = \text{election public key } (pk)$$

The \mathcal{VV}_{pk} function corresponds to the Cramer et al. ballot validity proof [14]. It is used to ensure that the u component of the candidate vote ($BitEnc(b)$) is in fact the encryption of $b' = 1$ or $b' = -1$, i.e. it is a valid candidate vote. The function outputs true if the candidate vote is valid and false if it is invalid. In the simplified MP protocol of section 2.1, this function can be used immediately after the first phase of the protocol, to ensure the correctness of the vote as soon as possible, or at the end of the voting process in phase 3.

Receipt Validity

$$\mathcal{RV}_{pk}(BitEnc(b), c, \vartheta, \omega) = validity$$

Where:

$$BitEnc(b) = \langle u, v \rangle$$

$$validity = \begin{cases} True & \text{if } \phi = \mathcal{E}_{pk}(c, \omega) = u^{c-\vartheta} \cdot v \\ False & \text{otherwise} \end{cases}$$

The receipt validity function corresponds to the zero knowledge verification code ϑ validation, which can be conducted by any trusted third party to prove that

$u^{c-\vartheta} \cdot v$ is the encryption of c , without any special knowledge but the public values. This is possible by a reconstruction of the c encryption using the ω encryption factor, revealed by the \mathcal{RC}_{pk} function. From equation [11](#), proving that $u^{c-\vartheta} \cdot v = \mathcal{E}_{pk}(c, \omega)$ is enough to prove that $\vartheta = \theta$ if $b' = 1$ or that $\vartheta = 2 \cdot c - \theta$ if $b' = -1$, which is enough to complement the voter's verification, unless $c = \theta$ where in both cases $\vartheta = \theta$. However, since c and θ are chosen randomly from \mathbb{Z}_q , this case probability is negligible. The \mathcal{RV}_{pk} function is used in the 3rd phase of the simplified MP vote protocol presented in section [2.1](#).

An alternative to the encryption reconstruction method, it is possible to prove that $u^{c-\vartheta} \cdot v = \mathcal{E}_{pk}(c, \omega)$ is an encryption of the value c without revealing ω using the Chaum and Pedersen protocol for proving the equality of discrete logarithms [12](#). The Chaum and Pedersen protocol can be used to prove the encryption $\mathcal{E}_{pk}(c, \omega) = \langle x, y \rangle$ proving that $\log_g x = \log_h y / (g^c)$.

Canonicalization

$$\mathcal{C}_{pk}(\text{BitEnc}(b), c, \vartheta) = \langle \text{canonicalVote} \rangle$$

Where :

$$\text{BitEnc}(b) = \langle u, v \rangle \wedge \text{canonicalVote} = u$$

The MP3 candidate vote canonicalization is very simple because the u element of the $\text{BitEnc}(b)$ is already an encryption of a fixed value, $b' \in \{-1, 1\}$, that depends on the candidate vote type, i.e. $\forall \text{BitEnc}(1) : b' = 1$ and $\forall \text{BitEnc}(0) : b' = -1$. Therefore, in MP3 the u encryption can be used directly as the *canonicalVote* because after an anonymization process, e.g. mix-net anonymization, it can be safely decrypted without revealing any link to the voter, i.e. the only thing we will see are the 1 and -1 values. MP1 and MP2 require a slightly more complex vote canonicalization.

4.1 Homomorphic Vote Tally

If the vote protocol uses the vote validity proof (i.e. ensures that all votes either encrypt a 1 or a -1) it is possible to use the efficient homomorphic vote tally process of the Cramer et al. vote protocol [14](#), since the MP3 candidate vote encryption u is equal to its vote encryption construction. Therefore, instead of decrypting each vote before counting it, which requires a previous anonymization process for each vote (usually using a mix-net), the MP3 homomorphic counting process performs the homomorphic addition of every encrypted vote $\text{vote}^j = \text{BitEnc}(b)_1^j \parallel \text{BitEnc}(b)_2^j \parallel \dots \parallel \text{BitEnc}(b)_k^j$, where $\text{BitEnc}(b)_i^j = \langle u_i^j, v_i^j \rangle, j = 1..n$, n is the number of valid votes and k is the number of candidates in each vote. Given that the vote validity function \mathcal{VV}_{pk} ensures that each $u_i^j = \mathcal{E}_{pk}(1)$ or $u_i^j = \mathcal{E}_{pk}(-1)$, then the vote counting for candidate i will be $\text{count}_i = \frac{n+d_i}{2}$, where d_i is the decryption of the homomorphic addition $\bigoplus_{j=1}^n u_i^j$. However, to ensure democracy, the protocol must also guarantee that each vote is counted for only one candidate, which means that the system must ensure that there is only

Table 1. MarkPledge functions computational costs in $mod p$ exponentiations. The MP2 $voteValidity$, $\mathcal{V}\mathcal{V}_{pk}$ and \mathcal{C}_{pk} values reflect our adjustments to the MP2 solution, cf. [18]. The MP2 matrix exponentiation, in $mod q$, is denoted as me .

	$\mathcal{V}\mathcal{E}_{pk}$ [BitEnc(b)] + [voteValidity]	$\mathcal{R}\mathcal{C}_{pk}$	$\mathcal{V}\mathcal{V}_{pk}$	$\mathcal{R}\mathcal{V}_{pk}$	\mathcal{C}_{pk}
MP3	[5] + [5]	0	8	5	0
MP1	[4 · α] + [-]	0	-	2 · α	$\approx \frac{\alpha}{2}$
MP1a	[2 + 4 · α] + [5]	0	8 + 2 · α	2 · α	0
MP2	[6 + 1 · me] + [8 + 1 · me]	1 · me	8	8 + 1 · me	3 + 1 · me

one $u_i^j = \mathcal{E}_{pk}(1)$ in each vote. Once again, given that each u_i^j is the encryption of the value 1 or -1, it is only necessary to prove that $\bigoplus_{i=1}^k u_i^j = \mathcal{E}_{pk}(2-k)$, e.g. using the Chaum and Pedersen protocol for proving the equality of discrete logarithms [11] or by revealing the sum of the encryption factors of the u_i^j elements, as suggested for the validation of the c encryption in the $\mathcal{R}\mathcal{C}_{pk}$ and $\mathcal{R}\mathcal{V}_{pk}$ functions.

4.2 Adjusting the Voter’s View of MP3 Output to the α Parameter

Usually, the MP security parameter α is set to a value between 20 and 30, which means that the voter must compare 4 to 5 character strings to verify that pledge = ϑ . In MP3 the c , ϑ and θ domains, and consequently the pledge domain, are defined by the cryptosystem parameter q and not by α . Since the size of q is in the hundreds of bits range we clearly have a usability issue. To solve this usability issue we propose a change in the voter’s view of the MP3 functions output, namely the voter’s view of the verification code ϑ and pledge value should be truncated to α bits by applying the $mod 2^\alpha$ operation to the referred values. Assuming an uniform and random distribution of ϑ and θ over \mathbb{Z}_q , the voter verification has a statistical soundness of $1 - 2^{1-\alpha}$, just because $q \gg 2^\alpha$, i.e. the voter still performs the verification of a random value uniformly distributed over \mathbb{Z}_{2^α} , cf. [18]. The soundness is $1 - 2^{1-\alpha}$ and not $1 - 2^{-\alpha}$ due to the case where $c = \theta$, where nothing is proved about the value b' encrypted in u .

5 Evaluation

This section discusses the improvements of MP3 over previous MP proposals in terms of each of the described functions. We first give a theoretical comparison of the techniques and then present the times for real implementations on smart cards. Note that by implementing MP in smartcards we may provide to each voter her own mobile voting machine without forcing her to trust any hardware/software device.

Table 1 shows the computational cost of the different MP functions in terms of the number of exponentiations per function. The values for MP3 are accordingly to the MP3 functions definition of section 4, and the MP1, MP1a and MP2

values are accordingly to the correspondent functions definition in [18]. All MP solutions use the exponential ElGamal encryption, which requires 3 exponentiations. Although, when the message to encrypt is a constant the corresponding message exponentiation was not considered. This is the case for all ElGamal encryptions in MP1 and MP1a and for the u encryption of MP3.

As detailed in [18], in MP1 each candidate encryption corresponds to α pairs of ElGamal encryptions, which corresponds to $4 \cdot \alpha$ exponentiations. The MP1 \mathcal{RV}_{pk} function performs α 1-out-of-2 cut-and-choose verifications, one for each of the α encryption pairs, which correspond to $2 \cdot \alpha$ exponentiations. The MP1 \mathcal{C}_{pk} function consists in homomorphically inverting the value inside approximately $\alpha/2$ ElGamal encryptions.

The MP1a row in table 1 represents the variant of MP1 proposed by Adida in [1]. MP1a adds an extra ElGamal encryption to each candidate vote, which allows the use of the *voteValidity* proof and verification proposed for MP3 in section 4. The MP1a \mathcal{VV}_{pk} function also needs to attest that the extra encryption “match” each one of the base α MP1 encryption pairs, cf. [1, 18].

The MP2 solution, cf. [4, 18], encrypts each candidate vote as the encryption of a 2D vector, which corresponds to 2 ElGamal encryptions, one for each of the vector coordinates. To compute the 2D vector it is necessary to perform a modular matrix exponentiation in q . The matrix exponentiation costs turned out to have a huge impact on the MP2 real performance, cf. section 5.1, thus they were also included in table 1. The MP2 *voteValidity* proof uses the same technique of MP1a and MP3, but needs some homomorphic vector algebra on the candidate vote encryption. The \mathcal{RC}_{pk} , \mathcal{RV}_{pk} and \mathcal{C}_{pk} costs also reflect the need of homomorphic vector algebra.

Excluding the matrix exponentiation, both MP2 and MP3 present an α order improvement over MP1(a) on the \mathcal{VV}_{pk} and \mathcal{RV}_{pk} functions, and on the candidate vote *BitEnc*(b) creation. In the \mathcal{C}_{pk} function both MP1a and MP3 are clearly better as they do not require any computation. The \mathcal{RC}_{pk} implementation on all MP versions is very simple and do not require any complex operation, except in MP2 where it is required a matrix exponentiation. Even excluding the matrix exponentiation operations, the improvements of MP3 over MP2 in the \mathcal{VE}_{pk} and \mathcal{RV}_{pk} functions are noteworthy. In the \mathcal{VE}_{pk} MP3 presents an 17% improvement in number of exponentiations that grows up to 28.5% with the vote validity, and in the \mathcal{RV}_{pk} MP3 presents an improvement of 37.5%.

When compared in terms of disk usage, MP3 and MP2 present approximately an α factor improvement over the MP1 and MP1a. In MP3 and MP2 a *BitEnc*(b) is composed by two ciphertexts, while in MP1 and MP1a a *BitEnc*(b) is composed respectively by $2 \cdot \alpha$ and $1 + 2 \cdot \alpha$ ciphertexts.

5.1 Implementation Results

In order to verify the real performance impact of MP3 we have partially implemented all MP solutions (MP1, MP2 and MP3). The two main reasons that lead us to the implementation were: i) the difficulty to compare the matrix exponentiation (required by MP2) to the large integer modular exponentiations needed for

the ElGamal encryptions, which are the base of all MP solutions, and ii) the curiosity to see if any MP solution can actually be run on limited devices, such as smart cards or secure elements inside a mobile phone (usually also implemented as smart cards) which can open the door for new vote protocols based on the MP technique.

We have implemented the matrix exponentiation using the non-recursive exponentiation by squaring algorithm (“computation by powers of 2”) described in [27]. We have optimized the matrix multiplication algorithm to take advantage of the special form of the $SO(2, q)$ matrices used by MP2. The $SO(2, q)$ test matrices, as defined by MP2, have elements in \mathbb{Z}_q and are exponentiated to random exponents in $\mathbb{Z}_{q\pm 1}$ (note that in MP2 the α bits vector indexes must be transformed into the corresponding exponents, which are uniformly distributed in $\mathbb{Z}_{q\pm 1}$, cf. [4,18]). We have tested the matrix exponentiation both on a PC and on smart cards (JavaCard and MULTOS). In the PC we have implemented the algorithm in Java. Our test code is available on request.

Our implementation results show that on a modern computer (Intel i5 2400 3.1GHz CPU) the matrix exponentiation time for $|q| = 160$ is about the same of an integer modular 1024 bits exponentiation with a 160 bits exponent. Thus, the \mathcal{VE}_{pk} function in MP3 presents an improvement of about 40% when compared to MP2. Moreover, the MP3 \mathcal{RV}_{pk} and \mathcal{C}_{pk} functions together, which are necessary to validate the election public data, are about 2.6 times faster than MP2. In other words, if with MP3 the election data verification takes one day it would take two and a half days with MP2.

Again, in a modern computer, the voter’s perception of the different vote encryption times would be close to none, as a large integer modular exponentiation takes only a few milliseconds. However, our implementation shows that the same is not true for more limited devices, e.g. smart cards or secure elements inside a mobile phone (also implemented as smart cards). We have partially implemented the different MP solutions in two different smart cards technologies: a JavaCard v2.2.1 (JCOP 31 v2.2) and a MULTOS v4.2.1 smart card (MC1-36K-61). The JavaCard technology was selected because it is widely deployed and it is being integrated as the secure element in many mobile phones; however the JavaCard API for large integer modular arithmetic is very limited, which has a negative impact on the MP performance. Thus, we also use a MULTOS card with a full large integer modular arithmetic native API.

Due to the very restricted large integer modular JavaCard 2.2.1 API (only modular exponentiation is available through the RSA engine) we had to program the modular addition, subtraction and multiplication operations. The addition and subtraction operations were implemented exclusively in “software”. On the other hand, the pure software approach for the modular multiplication was not viable. Thus, we use the formula $(a + b)^2 - (a - b)^2 = 4 \cdot a \cdot b$, as in [26]. This allowed us to perform the modular multiplication in an acceptable time, with the help of the JavaCard cryptographic processor; however it restricts the modulus size to values equal or above 512 bits. The “new” JavaCard 3 API still does not provide a large integer modular arithmetic support. On the other hand, the MULTOS card has all modular operations available in its native API.

Table 2. \mathcal{VE}_{pk} times in a NXP JCOP 31 v2.2 (JavaCard v2.2.1) with parameters $|p| = 1024$, $|q| = 512$ and $\alpha = 24$

JCOP 31 v2.2 (JavaCard) MarkPledge \mathcal{VE}_{pk} times ($ p = 1024$, $ q = 512$, $\alpha = 24$)			
	MP1a	MP2	MP3
<i>BitEnc(b)</i>	44.2 sec (1921%)	45 min (117391%)	2.3 sec (100%)
voteValidity	6.5 sec (100%)	45 min (41538%)	6.5 sec (100%)
Total	50.7 sec (576%)	90 min (30681%)	8.8 sec (100%)

Table 3. \mathcal{VE}_{pk} times in a MULTOS v4.2.1 MC1-36K-61 smart card with parameters $|p| = 1024$, $|q| = 512$ and $|q| = 160$, and $\alpha = 24$

MC1-36K-61 (MULTOS) MarkPledge \mathcal{VE}_{pk} times ($ p = 1024$, $ q = 512$, $\alpha = 24$)			
	MP1a	MP2	MP3
<i>BitEnc(b)</i>	29.4 sec (1729%)	1.5 min (5294%)	1.7 sec (100%)
voteValidity	2.6 sec (100%)	1.5 min (3462%)	2.6 sec (100%)
Total	32.0 sec (744%)	3.0 min (4186%)	4.3 sec (100%)

MC1-36K-61 (MULTOS) MarkPledge \mathcal{VE}_{pk} times ($ p = 1024$, $ q = 160$, $\alpha = 24$)			
	MP1a	MP2	MP3
<i>BitEnc(b)</i>	22.8 sec (1900%)	8.2 sec (683%)	1.2 sec (100%)
voteValidity	1.6 sec (100%)	8.8 sec (550%)	1.6 sec (100%)
Total	24.4 sec (871%)	17.0 sec (607%)	2.8 sec (100%)

Tables 2 and 3 present the performance times of the MarkPledge \mathcal{VE}_{pk} function on a smart card support. We only present times for this primitive as it is the only intensive cryptographic function that must be performed by the vote encryption device. The results for MP3 in Table 2 are real whilst for MP1a and MP2 were lower estimated. Table 3 presents real results for all MP versions.

The JavaCard times, in table 2, for the MP1a and MP2 are estimated as follows: the MP1a *BitEnc(b)* generation time is estimated as 80% of $\alpha \cdot \langle MP3 \textit{BitEnc}(b) \textit{time} \rangle$. Each of the MP1a α encryption pairs is composed of 2 ElGamal encryptions, just like the *BitEnc(b)* of MP3. The 80% adjustment comes from the fact that it takes one less exponentiation because it encrypts two constant values, cf. [11, 18, 22] and table 1. The MP1a *voteValidity* generation time is equal to the MP3 *voteValidity* generation time, since both use exactly the same technique. The MP2 *BitEnc(b)* generation time is estimated as $\langle MP3 \textit{BitEnc}(b) \textit{time} \rangle + \langle \textit{matrix exponentiation time} \rangle$. The MP2 *BitEnc(b)* is also composed of 2 ElGamal encryptions, however the values to encrypt are the result of a matrix exponentiation and therefore this time must be added to the encryption time. The MP2 *voteValidity* (as we have specified in [18]) needs another matrix exponentiation and three integer exponentiations to enable the use of the vote validity technique proposed for MP3 and MP1a. Therefore, we use

$\langle MP3 \text{ voteValidity time} \rangle + \langle \text{matrix exponentiation time} \rangle$ as a lower estimative for the $MP2 \text{ voteValidity time}$.

As can be easily seen in tables [2](#) and [3](#), the use of any non native cryptographic function comes at a huge cost. In the best scenario MP2 only presents a 30% improvement over MP1 and is 6 times worst than MP3.

Note that the results are for one invocation of the \mathcal{VE}_{pk} function, i.e. the results are for each individual candidate vote encryption. Thus, the values presented must be multiplied by the number of candidates to give the full vote encryption time. In practice, a full 10 candidate ballot encryption with proofs takes on the JavaCard 8.45 minutes, 15 hours or 1.47 minutes using respectively MP1a, MP2 and MP3. The results for the MULTOS card, with the same setup, are respectively 5.3 minutes, 30 minutes and 43 seconds. With a more standard parameters setup ($|p| = 1024$ and $|q| = 160$), the results for the MULTOS card are respectively 4 minutes, 2.8 minutes and 28 seconds.

6 Conclusions

This paper presented the MP3 specification, which is more efficient and simpler than all other previous MP solutions. Moreover, our implementation tests show that, in low computational power devices, e.g. smart cards or secure elements inside a mobile phone (usually also smart cards), only MP3 presents an acceptable performance. The MP3 performance improvements are mainly due to its simple mathematical structure. That simple structure comes with a small price: the MP3 soundness is $1 - 2^{1-\alpha}$ vs the $1 - 2^{-\alpha}$ soundness of the previous MP solutions.

References

1. Adida, B.: Advances in Cryptographic Voting Systems. Ph.D. thesis, MIT (August 2006)
2. Adida, B.: Helios: Web-based open-audit voting. In: 17th USENIX Security Symposium (2008)
3. Adida, B., Neff, A.: Ballot casting assurance. In: EVT 2006. USENIX/ACCURATE, Vancouver, B.C. (2006)
4. Adida, B., Neff, A.: Efficient receipt-free ballot casting resistant to covert channels. In: EVT/WOTE 2009. USENIX/ACCURATE/IAVOSS, Montreal, Canada (August 2009)
5. Benaloh, J.: Simple verifiable elections. In: EVT 2006. USENIX/ACCURATE, Vancouver, B.C. (2006)
6. Benaloh, J.: Ballot casting assurance via voter-initiated poll station auditing. In: EVT 2007. USENIX/ACCURATE, Boston, MA (2007)
7. Benaloh, J.C.: Verifiable Secret-Ballot Elections. Ph.D. thesis, Yale University (1987)
8. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM* 24(2), 84–88 (1981)
9. Chaum, D.: Secret-ballot receipts: True voter-verifiable election. *IEEE Security & Privacy* 02(1), 38–47 (2004)

10. Chaum, D.: Punchscan (September 2009), <http://www.punchscan.org/>
11. Chaum, D., Pedersen, T.P.: Wallet Databases with Observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)
12. Chaum, D., Ryan, P.Y.A., Schneider, S.: A Practical Voter-Verifiable Election Scheme. In: De Capitani di Vimercati, S., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 118–139. Springer, Heidelberg (2005)
13. Clarkson, M., Chong, S., Myers, A.: Civitas: Toward a secure voting system. In: IEEE Symposium on Security and Privacy, pp. 354–368 (May 2008)
14. 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)
15. ElGamal, T.: A public-key cryptosystem and signature scheme based on discrete logarithms. IEEE Transactions on Information Theory IT-31(4), 469–472 (1985)
16. Fujioka, A., Okamoto, T., Ohta, K.: A Practical Secret Voting Scheme for Large Scale Elections. In: Seberry, J., Zheng, Y. (eds.) AUSCRYPT 1992. LNCS, vol. 718, pp. 244–251. Springer, Heidelberg (1993)
17. 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)
18. Joaquim, R., Ribeiro, C.: An efficient and highly sound voter verification technique and its implementation - extended version. Tech. Rep. 40/2011, INESC-ID (September 2011)
19. Joaquim, R., Ribeiro, C., Ferreira, P.: VeryVote: A Voter Verifiable Code Voting System. In: Ryan, P.Y.A., Schoenmakers, B. (eds.) VOTE-ID 2009. LNCS, vol. 5767, pp. 106–121. Springer, Heidelberg (2009)
20. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: WPES, Alexandria, Virginia, USA, pp. 61–70 (November 2005)
21. 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), <http://www.seas.harvard.edu/~talm/papers/MN06-voting.pdf>
22. Neff, C.A.: Practical high certainty intent verification for encrypted votes (2004), <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.134.1006&rep=rep1&type=pdf>
23. NIST: Digital signature standard (dss) (June 2009), http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf, FIPS 186-3
24. 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)
25. Sandler, D., Derr, K., Wallach, D.S.: Votebox: A tamper-evident verifiable electronic voting system. In: 16th USENIX Security Symposium (2007)
26. Sterckx, M., Gierlichs, B., Preneel, B., Verbauwhede, I.: Efficient implementation of anonymous credentials on java card smart cards. In: 1st IEEE International Workshop on Information Forensics and Security, pp. 106–110 (2009)
27. Wikipedia: (April 2011), http://en.wikipedia.org/wiki/Exponentiation_by_squaring#Computation_by_powers_of_2

Single Layer Optical-Scan Voting with Fully Distributed Trust^{*}

Aleksander Essex¹, Christian Henrich², and Urs Hengartner¹

¹ Cheriton School of Computer Science
University of Waterloo
Waterloo, ON, Canada N2L 2G1
{aessex,uhengart}@cs.uwaterloo.ca

² Institut für Kryptographie und Sicherheit/EISS
Karlsruhe Institute of Technology
76128 Karlsruhe, Germany
christian.henrich@kit.edu

Abstract. We present a new approach for cryptographic end-to-end verifiable optical-scan voting. Ours is the first that does not rely on a single point of trust to protect ballot secrecy while *simultaneously* offering a conventional single layer ballot form and unencrypted paper trail. We present two systems following this approach. The first system uses ballots with randomized confirmation codes and a physical in-person dispute resolution procedure. The second system improves upon the first by offering an informational dispute resolution procedure and a public paper audit trail through the use of self-blanking invisible ink confirmation codes. We then present a security analysis of the improved system.

1 Introduction

Research into cryptographically “end-to-end” verifiable *optical-scan* voting systems has come a long way toward practicality. This progress has not come easily: academics and election administrators often struggle to agree on a vast and often orthogonal set of core system properties. Similar in spirit to Benaloh [2], we advocate the *coexistence* of modern cryptographic proofs of correctness and conventional, lower-tech, methods for auditing elections. In this paper we tackle a long standing trade-off of properties in the voting literature: distributed trust versus a conventional optical-scan paper ballot form.

Typically cryptographic voting schemes allow the voter to construct a receipt of their vote enabling each voter to confirm the inclusion of their ballot in the election tally. In order to protect ballot secrecy, the association between a receipt and the corresponding (clear-text) vote must be kept hidden at all times. Many proposals have relied on trusted entities or hardware to enforce this, especially with regards to ballot printing. Other proposals distribute trust among multiple entities through the use of specialized multi layer ballot forms.

* Full version available: <http://eprint.iacr.org/2011/568>

Our Proposal. We consider a list of requirements for end-to-end verifiable optical scan voting that factors a diverse set of stakeholders (i.e., cryptographers, election officials, legislators, democracy groups, etc). This list is by no means exhaustive and does not encompass challenges faced by other voting methods (e.g., internet, mail-in, etc). Our list is as follows:

1. **Distributed trust:** No single party, *including* the ballot printer(s), gains an advantage in deducing how a voter voted or in linking a receipt to its corresponding clear-text vote. This is a vital requirement of any secret ballot election employing the receipt paradigm.
2. **Single layer ballot form:** A ballot is a single sheet of paper with a *fixed order* candidate list¹ and the voter marks the optical scan ovals *directly beside* their chosen candidate. Multi layer ballots are an artifact of cryptographic voting, requiring voters to re-learn how to cast a ballot. Our experience in running real-world cryptographic elections—both with single layer and with multi layer ballot forms—has indicated to us that multi layer ballots are more cumbersome for voters and more difficult to administer for election officials [14,5,41].
3. **Human-readable paper audit trail:** Pursuant to the legal requirements of many jurisdictions voting, voting intent remains plainly evident on cast ballot forms. Such an audit trail also allows for recoverability in the event of lost or forgotten cryptographic keys or other unforeseen errors.
4. **Public paper audit trail:** The collection of cast ballot forms (i.e., the *paper audit trail*) can be made public without revealing the link between receipt and clear-text vote. A public audit paper trail may also be a legal requirement and is critical in protecting ballot secrecy during a manual recount.

In this paper we propose two novel end-to-end verifiable optical scan voting systems that meet all four of these requirements. Some of these properties have been examined in the literature, but no proposal has achieved all of them. Scantegrity achieves 2 and 3 [9,7]. Prêt à Voter and Scratch & Vote achieve 2 and 4 [10,37,142], two Punchscan variants achieve only 4 [19,22], and each of Split-Ballot Voting, ClearVote and Kusters *et al.* achieve 1 and 4 [28,34,23]. A proposal due to Benaloh [2] achieves 2, 3, and 4. See the section on related work for additional discussion.

Contributions. We present two novel systems for single layer optical-scan voting with distributed trust based respectively on the ballot styles used by Scantegrity [9] and Scantegrity II [7].

Basic System: We propose a basic two-party system for creating ballot forms with randomized confirmation codes that meets properties 1, 2, and 3. It relies on a private paper audit trail and an in-person physical dispute-resolution procedure.

¹ There are also potential advantages to using ballots with randomized candidate lists. Our system can accommodate this approach with minor protocol changes.

Improved System: We then propose an improved two-party system that uses ‘self-blanking’ invisible ink confirmation codes. It improves on the basic system by allowing the paper audit trail to be made public, thereby achieving all four properties. In addition it offers an *informational* dispute-resolution procedure allowing disputes to be resolved based on knowledge of a confirmation code (as opposed to physical possession of a receipt).

2 Preliminaries

2.1 Physical Primitives

End-to-end verifiable ballots often employ physical security methods as part of the receipt creation process. The use of physical security mechanisms can be contentious due to inherent questions regarding their cost, feasibility, and real-world security properties. However, there is precedent for protocols built around *ideal* physical security mechanisms (c.f. [18,27]). Throughout the rest of this paper we assume that all physical security mechanisms function ideally. Broadly speaking the ballot secrecy properties of our systems reduce to those of Scantegrity’s when the physical security mechanism fail. A brief discussion of this is included in our security analysis in the full paper. ²

Physical Security Mechanisms. We briefly summarize the physical security mechanisms employed by our systems.

Invisible ink as its name implies is initially invisible when printed and becomes visible only after *activation*. It was proposed for use in the Scantegrity II system [7], and has been implemented and fielded in a live municipal election in the United States [5]. For the improved system presented in Section 4 we additionally make use of a ‘slow’ developing ink,

Scratch-off coating is a convenient, cost-effective and widely available method for concealing (and subsequently revealing) printed information. It has been employed in several voting schemes (cf. [1,39]) to protect ballot secrecy,

Visual cryptography [29] is a well known technique for visually implementing a logical exclusive disjunction (i.e., an *xor*) built from a physical medium acting as a logical disjunction (i.e., an *or*). A message or graphical image can be split into two or more information-theoretically secure shares. When the shares are combined (i.e., overlaid) the message becomes visually perceptible.

Physical Security Sub-protocols: We briefly summarize the physical security sub-protocols used by our systems.

² <http://eprint.iacr.org/2011/568>

Document Authenticity: We require a method for determining a document’s authenticity. Classical methods for anti-counterfeiting (e.g., watermarks, holographic foil, embedded magnetic strips, etc) can be cost-prohibitive. Paper fibre analysis (cf. [12]) using commercial-grade scanners is possible³. For the sake of our description we assume that there exists an efficient physical scheme for determining a ballot’s authenticity,

Private Printing: we make use of private printing techniques to pick and print *human-readable* confirmation codes on ballots without either printer individually knowing which codes were printed. A proposal for two-party private printing was made in [15]. Private printing is used in the improved system.

2.2 Cryptographic Primitives

We briefly outline the main cryptographic primitives used by our systems. We note that these primitives are standard across the cryptographic voting literature.

Homomorphic Encryption. Let $\langle \text{DKG}, \text{Enc}, \text{DDec} \rangle$ be a *distributed public-key encryption* scheme. Without loss of generality, DKG generates two private key shares x_1 and x_2 for parties \mathcal{P}_1 and \mathcal{P}_2 respectively and a joint public key Y . Encryption $\llbracket m \rrbracket = \text{Enc}_Y(m, r)$ is semantically secure and homomorphic in at least one operation. Decryption $m = \text{DDec}_{(x_1, x_2)}(\llbracket m \rrbracket)$ requires both key shares. Specifically we will make use of exponential Elgamal [13] with distributed decryption [33]. For simplicity we will omit the public-key when implied. We additionally require a *partially-homomorphic xor* operation $\tilde{\oplus}$ such that, for a pair of messages $m_1, m_2 \in \{0, 1\}$, $\llbracket m_1 \rrbracket \tilde{\oplus} m_2$ produces a ciphertext that encrypts the bitwise xor of the associated plaintext bits, i.e., $\llbracket m_1 \oplus m_2 \rrbracket$. We present a bit encryption scheme based on exponential Elgamal in the full paper² though there is more than one way to accomplish this (cf. [20,30]).

Mixnets. Mixnets have long been a fixture in cryptographic voting. We make use of a simple re-encryption mixnet (cf. [31]) structure to create our proofs (we do not utilize a separate proof of correct mixing, as it is provided by other parts of our system). *Re-randomization* (a.k.a., re-encryption) of a ciphertext c is accomplished by computing $c' = \text{ReRand}(c, r) = c \cdot \text{Enc}(0, r)$ ⁴. By rerandomizing and shuffling a batch of ciphertexts we implement a simple *reencryption mixnet*, Mix. In this paper, when applying Mix to a matrix of ciphertexts, we describe mixing as occurring on *tuples* of ciphertexts grouped by columns and shuffled by rows.

Commitments. We use a cryptographic *commitment* scheme to commit to permutations as part of a cut-and-choose proof of shuffle. The dispute resolution

³ In general there are privacy threats due to fingerprinting documents however this is not a threat to ballot secrecy assuming non-collusion.

⁴ Replace 0 with the *identity* element for other groups.

procedure in the improved system requires the prover to either unveil (i.e., *de-commit* to) the code, or alternatively to issue a *non-interactive proof of plaintext inequality*. A commitment inherent to IND-CPA secure encryption fits this dual role. Here a sender *commits* to a message m by posting its encryption $\llbracket m \rrbracket = \text{Enc}(m, r)$. Later the commitment can be unveiled when the sender reveals an m', r' , allowing anyone to verify $\text{Enc}(m', r') = \llbracket m \rrbracket$, and hence $m' = m$. This approach is commonly used in several voting schemes (e.g., [3,11,40]).

Non-interactive Challenges. As part of our cut-and-choose correctness proof we require a method for fairly generating random challenge bits. Loosely speaking, *fairness*, requires that no one is able to predict, or controllably influence the output with non-negligible advantage. Furthermore, the fairness of the method should be *convincing* to voters. Both the heuristic due to Fiat and Shamir [17], and the notion of a *random beacon* (cf. [36,11]) are possibilities.

2.3 Participants

There are several entities that participate in the election.

- A set of **voters** with the authority to cast a ballot in the election, optionally construct a privacy-preserving receipt of their vote, and optionally participate in an election audit,
- An **election operations commission** \mathcal{C} with the capability and authority to organize and run an election, operate a polling place, optically scan ballots, report results, act as a custodian of the cast ballot record, and participate in an in-person dispute resolution procedure,
- Two independent **ballot printers** $\mathcal{P}_1, \mathcal{P}_2$ who possess the capability and authority to print documents in the untrusted printing model and participate in a secure (cryptographic) two-party computation,
- An election **scrutineer** \mathcal{S} with the authority to audit the correctness of printed ballots relative to their cryptographic representation. Additionally \mathcal{S} acts as a proxy for voters during disputes with \mathcal{C} to protect their identity. In practice there might be any number of election auditors, representing the candidates or other democracy groups.

As a fundamental requirement of our security model, we assume that neither printer nor election commission collude with one another.

3 The Basic System

The basic system produces a public and universally verifiable cryptographic proof attesting to the correctness of the election’s outcome. This correctness proof is based on standard cut-and-choose techniques (cf. [9,7,8]). Without loss of generality we consider a single-contest election involving n ballots⁵ and m candidates. The basic system involves several protocols. The protocols `generateBallots`,

⁵ The number of ballots printed is the total number of voters times a *heuristically* chosen expansion factor to account for audited and spoiled ballots.

Table 1. Notations

n	Number of ballots to print	T	List of all ballot-tuples
m	Number of candidates	BallotTable	Table of ballot information
d	Bit-length of ballot-id	ReceiptTable	Table of receipt information
L	List of candidate names	MP_1/MP_2	Printer 1/2's master permutation
Σ	Confirmation code alphabet	π/ρ	Random perm'ns composing to MP_1
α	Soundness parameter	σ/τ	Random perm'ns composing to MP_2
b/B	Ballot-id/list of ...	MidMarks	Intermediate mark state list
r/R	Receipt-id/list of ...	MidMarksP1	\mathcal{P}_1 's intermediate mark state list
c/C	Confirmation code/list of ...	MidMarksP2	\mathcal{P}_2 's intermediate mark state list
μ	Mark-state of opscan oval	eid	Election-unique identifier

`preElectionPrep`, `postElectionPrep` encompass the preparation for the public election audits. Note that each of these protocols taken individually is only secure in an *honest-but-curious* setting. To make them robust against an active adversary we make use of a set of *audit* protocols `proveScan`, `proveReceipt`, `provePrinting` and `resolveDispute`. A summary of notations used is presented in Table 1.

The Ballot. The basic optical-scan paper ballot form has a pre-printed, fixed-order candidate list $L = \{l_1 \dots l_m\}$. Adjacent to each candidate is an optical scan oval with a *mark state* $\mu \in \{0, 1\}$ corresponding respectively to whether the oval was unmarked or marked. The ballot form is separated into two regions by a perforation. The top constitutes the *ballot portion*, and the bottom is the *receipt portion*. An alphabet Σ of m confirmation codes is defined. Each optical scan oval (and hence each candidate) is associated with a confirmation code drawn independently at random, and without replacement, from Σ . A *ballot-id* b is a d -bit⁶ vector printed on the ballot portion. An independent *receipt-id* r is printed on the receipt portion. The first printer prints the receipt-ids under a scratch-off coating and the second prints the confirmation codes. Both printers will jointly print the ballot-id in invisible ink. Printing of the ballot- and receipt-ids is done such that each printer only knows what *it* prints (and not what its counterpart prints). The basic ballot is depicted in Figure 1(a).

Ballot Tuple. A ballot is fully specified by the tuple $\{b, r, c\}$, which denotes the association between a unique *ballot-id* bit vector $b \in \{0, 1\}^d$, a unique *receipt-id* $r \in \{1 \dots n\}$, and a random permutation of *confirmation codes* $c = \pi(\Sigma)$ for a permutation π drawn independently and uniformly at random from the set of possible permutations of Σ .

⁶ Since in the basic scheme ballot-ids are the xor of random bit vectors, d is chosen to be large enough so as to make duplicate ballot-ids highly unlikely.

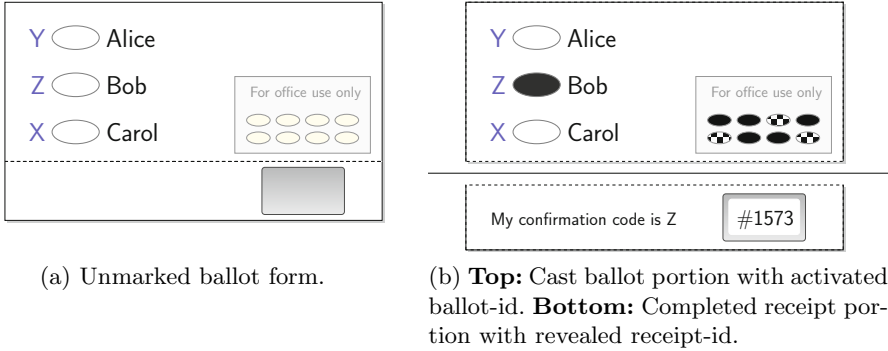


Fig. 1. Basic ballot: Optical-scan ballot form with *ballot portion* (top) and *tear-off receipt portion* (bottom) depicting a randomized confirmation code list, a unique ballot-id printed in *invisible ink visual-crypto* and a unique receipt-id beneath a scratch-off coating. Ballot printing is distributed between two printers such that neither can match receipts with cast ballots.

3.1 Election Preparation

The election is initialized as follows: election commission \mathcal{C} initializes a *public bulletin board* \mathcal{BB} ⁷ and a unique election identifier eid . Printers \mathcal{P}_1 and \mathcal{P}_2 jointly run DKG. They post the public key Y to public bulletin board \mathcal{BB} and retain their respective private key shares x_1, x_2 . This list of public parameters $\text{pubParam} = \{n, m, d, L, \Sigma, \alpha, eid, Y\}$ is posted to \mathcal{BB} . All functions/protocols accept pubParam as input.

Ballot Tuple Creation. The printers now jointly generate encrypted ballot tuples by running `generateBallots`. This protocol is given in Algorithm 1.

Ballot Printing. The n ballot forms are printed in three steps. For each ballot-tuple a paper ballot is prepared in the following order:

- **Static background:** directions, candidate names, etc, printed in black ink,
- \mathcal{P}_1 's **share:** the receipt-id is printed and concealed under scratch-off coating, \mathcal{P}_1 's share of the ballot-id printed in invisible ink visual-crypto,
- \mathcal{P}_1 's **share:** the confirmation codes are printed in regular ink, \mathcal{P}_2 's share of the ballot-id printed in invisible ink visual-crypto over \mathcal{P}_1 's share.

The completed ballot forms are then randomly shuffled and delivered into the custody of the election commission \mathcal{C} . Throughout the ballot printing and voting phases the printers will conduct random audits of ballot forms to ensure their

⁷ Typically modelled as an append-only broadcast channel with state (cf. [4]).

Algorithm 1. generateBallotsParticipants: Printers $\mathcal{P}_1, \mathcal{P}_2$

```

1 Printer  $\mathcal{P}_1$  should:
2   for  $i \in \{1 \dots n\}$  do
3     Encrypt vectors of random bits:
4      $B'(i) \leftarrow (\text{Enc}(\text{randBit}), \dots, \text{Enc}(\text{randBit}))$ 
5     Post a non-malleable commitment to each randBit along with the
      random factor used to encrypt it.
6   Encrypt and shuffle receipt-ids:
7    $R \leftarrow \text{Shuffle}(\text{Enc}(1) \dots \text{Enc}(n))$ 
8 end

9 Printer  $\mathcal{P}_2$  should:
10  for  $i \in \{1 \dots n\}$  do
11    Randomly shuffle and encrypt code confirmation codes:
12     $C(i) \leftarrow \text{Shuffle}(\text{Enc}(\Sigma(1)) \dots \text{Enc}(\Sigma(m)))$ 
13 end

14 Both Printers should:
15 | Simultaneously and respectively output  $B', R$  and  $C$  to  $\mathcal{BB}$ .
16 end

17 Printer  $\mathcal{P}_2$  should:
18  for  $i \in \{1 \dots n\}; j \in \{1 \dots d\}$  do
19    Homomorphically xor random bits:
20     $b'_1 \dots b'_d \leftarrow B'(i)$ 
21     $B(i) \leftarrow (b'_1 \tilde{\oplus} \text{randBit}, \dots, b'_d \tilde{\oplus} \text{randBit})$ 
22    Post a non-malleable commitment to each randBit along with the
      random factor used in computing the xor.
23  Output  $B$  to  $\mathcal{P}_1$ 
24 end

//Remark:  $\text{Shuffle}(X)$  applies a permutation to a list  $X$ , drawn independently
and uniformly randomly from the set of permutations of size  $|X|$ . randBit
returns a single bit drawn independently and uniformly at random. It is
possible that  $\mathcal{P}_2$  might attempt to maliciously select its bits as a function of
 $\mathcal{P}_1$ 's. However  $\mathcal{P}_2$  will not know (beyond a guess) what to print on the ballot,
and will be caught in ProvePrinting with statistical certainty.

```

authenticity and to look for signs of tampering (e.g, to catch if someone reveals the secret information then replaces the ballot with a replica). Note that if either printer prints something *other* than their contribution in generateBallots (e.g., if a printer prints an all-black VC pixel), this will be caught in provePrinting with statistical confidence dependent on the number of audited ballots.

Pre-election Proof Preparation. The printers initialize the public audit dataset and cut-and-choose correctness proofs by running preElectionPrep. This protocol is given in Algorithm [2](#).

Voting and Receipt Creation. An individual wishing to vote shall attend the polling place and authenticate themselves to \mathcal{C} . All qualified and authenticated individuals (i.e., voters) are then eligible to receive a ballot. The voter selects a ballot form at random from a stack of unmarked ballot forms and takes it, a regular (black) marking pen, and a privacy sleeve into a private voting booth. The voter marks the oval next to their preferred candidate l_i on the ballot portion. Then, if they so choose, the voter creates a receipt of their vote by noting the code letter c_i and writes it in the appropriate space on the receipt portion. The voter then places the marked ballot form into the privacy sleeve and returns it to the poll worker. The poll worker confirms the receipt-id’s scratch-off coating is still intact and the ballot-id has not been activated (rejecting the ballot in such a case), then detaches the receipt portion and places it on a table in view of the voter. The ballot portion is then fed into the optical scanner. If the ballot is accepted the receipt portion is retained by the poll worker. If the ballot portion is successfully cast, the receipt portion is returned to the voter and the voting process is complete. A diagram showing completed ballot and receipt portions is depicted in Figure 1(b).

A Note about Timing Attacks. In some jurisdictions, poll workers keep a poll book of voter identities in the *order they voted*. If the scanner were to likewise maintain the order of cast ballots it, taken along with the poll book, would compromise ballot secrecy. Since in our case the ballot is drawn at random from the pile, and the poll worker does not see the ballot- or receipt-ids, this threat can be mitigated by having voters cast ballots into a ballot box at the polling place and then scanning them later at a central location.

Post-election Proof Preparation. After the election \mathcal{C} populates the `BallotTable` with the mark state information collected by the optical scanners. With this data the printers and can now finalize the cut-and-choose correctness proof by running `postElectionPrep`. This protocol is given in Algorithm 3.

3.2 Audits

There are three simultaneous properties that must be proven in order for the overall results to be proven correct. These audits include,

- **Proving correct mark-state reporting by \mathcal{C} :** Using their receipt, a voter \mathcal{V} checks whether \mathcal{C} correctly registered their vote by running `proveScan`,
- **Proving mark-state propagation by $\mathcal{P}_1, \mathcal{P}_2$:** The printers prove to any interested party that they honestly applied their master permutations to mark state information in `BallotTable` by running `proveReceipt`,

Algorithm 2. preElectionPrep

Participants: Printers $\mathcal{P}_1, \mathcal{P}_2$ **Public Input:** Candidate list L **Private Input:** Lists of encrypted ballot-ids B , receipt-ids R , and code shuffles C **1 Both Printers should:***//Expand the n ballot tuples into a table of mn rows (one for every candidate on every ballot):***2 for** $i \in \{0 \dots n - 1\}$ **do****3** $c_1 \dots c_m \leftarrow C(i)$ **4** **for** $0 \leq j \leq m - 1$ **do****5** $T(1, mi + j) \leftarrow B(i)$ **6** $T(2, mi + j) \leftarrow \text{Enc}(L(j + 1))$ **7** $T(3, mi + j) \leftarrow R(i)$ **8** $T(4, mi + j) \leftarrow c_j$ *// \mathcal{P}_1 followed by \mathcal{P}_2 using master permutations MP1 and MP2 respectively:***9** $T' \leftarrow \text{Mix}(T)$ *//Create ballot and receipt tables:***10** BallotTable $\leftarrow \text{DDec}(T'(1 \dots 2, :))$ **11** ReceiptTable $\leftarrow \text{DDec}(\text{Mix}(T'(3 \dots 4, :)))$ **12** Post BallotTable, ReceiptTable to \mathcal{BB} **13 end***//Prepare cut-and-choose proof of correspondence between elements in the ballot and receipt tables:***14 Printer \mathcal{P}_1 should:****15 for** $i \in \{1 \dots \alpha\}$ **do****16** Choose $\pi_i \in_R \Pi_{mn}$ **17** Set ρ_i such that $\rho_i \circ \pi_i = \text{MP}_1$ **18** Post Commit(π_i), Commit(ρ_i) to \mathcal{BB} **19 end****20 Printer \mathcal{P}_2 should:****21 for** $i \in \{1 \dots \alpha\}$ **do****22** Choose $\sigma_i \in_R \Pi_{mn}$ **23** Set τ_i such that $\tau_i \circ \sigma_i = \text{MP}_2$ **24** Post Commit(σ_i), Commit(τ_i) to \mathcal{BB} **25 end***//Remark: Let $x \in_r \Pi_y$ denote a permutation function x drawn independently and uniformly at random from the set of permutations of list of y elements.**Let $\text{MP}_1, \text{MP}_2 \in_R \Pi_{mn}$. Then for $i \in \{1 \dots \alpha\}$, we have* *$\tau_i \circ \sigma_i \circ \rho_i \circ \pi_i = \text{MP}_2 \circ \text{MP}_1$.*

Algorithm 3. `postElectionPrep`

Participants: Election Commission \mathcal{C} , Printers $\mathcal{P}_1, \mathcal{P}_2$
Private Input: Secret Master permutations MP_1, MP_2 , Scanned Cast Ballots

//Populate BallotTable with scanner data

- 1 **Election commission \mathcal{C} should:**
- 2 **foreach** $\{b, s, \mu\}$ *recorded by scanner* **do**
- 3 Find i for which `ballotTable(1, i) = b`
- 4 and `ballotTable(2, i) = s`
- 5 `ballotTable(3, i) ← μ`
- 6 Post `ballotTable(3, :)` to \mathcal{BB} .
- 7 **end**

//Propagate marks from BallotTable to ReceiptTable

- 8 **Printer \mathcal{P}_1 should:**
- 9 `MidMarks ← MP_1 (BallotTable(3, :))`
- 10 Post `MidMarks` to \mathcal{BB} **for** $i \in \{1 \dots \alpha\}$ **do**
- 11 `MidMarksP1i ← π_i (BallotTable(3, :))`
- 12 Post `MidMarksP1i` to \mathcal{BB}
- 13 **end**
- 14 **Printer \mathcal{P}_2 should:**
- 15 `ReceiptTable(3, :) ← MP_2 (MidMarks)`
- 16 Post `ReceiptTable(3, :)` to \mathcal{BB} . **for** $i \in \{1 \dots \alpha\}$ **do**
- 17 `MidMarksP2i ← σ_i (MidMarks)`
- 18 Post `MidMarksP2i` to \mathcal{BB}
- 19 **end**

- **Proving printed ballot forms match \mathcal{BB} :** A scrutineer \mathcal{S} ⁸ runs `provePrinting` with the printers to verify that the ballot tuple information conveyed by the paper ballot forms *matches* the ballot tuple representation in \mathcal{BB} . Audited ballots are *spoiled* and not counted.

Because receipt creation is unsupervised, a dispute may arise between \mathcal{C} and \mathcal{V} over the correct confirmation code. In such an event a dispute resolution protocol can be run. For space reasons we defer complete listings of `proveScan`, `proveReceipt`, `provePrinting` and the dispute resolution procedure to the full paper.²

4 Improved System

In this section we present a system that improves upon the basic system in two ways: First, it replaces the physical dispute resolution procedure with an *informational* dispute procedure. Second, the collection of cast ballots (i.e., the paper audit trail) can be viewed publicly without compromising ballot secrecy.

⁸ A scrutineer is not strictly necessary. Voters themselves may choose to initiate this audit, although in our experience they rarely do!

Informational Dispute Resolution. The dispute resolution procedure of the basic system is inefficient and time consuming. Chaum et al. proposed the notion of invisible ink confirmation codes in Scantegrity II [7] as an *informational* means of resolving dispute. Under this approach, codes are printed in invisible ink, and only revealed to the voter if marked. Assuming the code space is sufficiently large so as to make successful random guess unlikely, then knowledge of *any* valid code can be taken as evidence that a voter correctly created their receipt. Any discrepancy found between a receipt and the ReceiptTable can then be attributed to \mathcal{C} (assuming the other correctness proofs are valid). In the improved system, we create and print the codes using a *private printing* protocol. Thus the role of invisible ink is twofold: it restricts the voter’s knowledge of unmarked codes *and* it prevents the printers from linking receipts to votes.

Public Paper Trail. Invisible ink confirmation codes require a code space that makes random guessing statistically unlikely. For example Scantegrity II proposes a 3-digit code (making a random guess successful 0.1% of the time on average). However in the presence of unique (or semi-unique) codes, access to cast ballots coupled with the public audit dataset is sufficient (or nearly sufficient) to allow *any* observer to link receipts to clear-text votes. This not only means that the paper ballot record must be kept *secret*, but further that the custodian of the ballot record (i.e., \mathcal{C}) is trusted with knowledge of how voters voted. This is one of the major limitations of Scantegrity II. To address this privacy weak-spot, we require a method for not only privately printing a confirmation code, but for displaying it *only while the voter is in the booth*. In the presence of “disappearing” codes, not only can we offer distributed trust with respect to $\mathcal{P}_1, \mathcal{P}_2$ and \mathcal{C} , but we can also make the paper ballot record public.

Self-blanking Confirmation Codes. We propose a method for printing of confirmation codes that is self-blanking (i.e., the message is only temporarily visible). The standard invisible ink described by Scantegrity II activates *instantaneously*. That is to say, the chemical reaction responsible for the ink’s pigmentation completes on the order of milliseconds. It was suggested in [7] that a *slower* reacting ink might be the addition of an *anti-catalyst*. This substance, if present, can slow down pigmentation by seconds or minutes (depending on design needs). Combining the technique of visual cryptography with such a ‘slow’ invisible ink, we can construct a self-blanking pixel (see Table 2). Finally, combining self-blanking pixels with the private printing protocol of [15], we can print confirmation codes that are both distributed between two-parties and self-blanking.

The Improved Ballot. The improved ballot differs from the basic ballot in that it makes use of *self-blanking invisible ink* confirmation codes. The codes are printed inside the optical scan ovals in *self-blanking invisible ink*. When the voter marks an oval using the specially provided activator pen, the confirmation code is revealed allowing the voter (finite) opportunity to write down the code

Table 2. Self-blanking VC Pixel. Two sub-pixels contain invisible ink. Each party applies an anti-catalyst (cyan) to *one* sub-pixel. Sub-pixels containing this substance darken more slowly than those without ($t = 0$ is the moment of activation). Eventually all sub-pixels darken “blanking” the pixel’s value.

a	b	VC		Result when activated		
		$VC(a)$	$VC(b)$	$t = 0$	$t > 0$	$t \gg 0$
0	0	■	∅	■	■	■
0	1	■	∅	■	■	■
1	0	∅	■	■	■	■
1	1	∅	■	■	■	■

on their receipt. Eventually the oval darkens completely indicating *that* the oval was chosen by the voter, but not what the confirmation code was (see Figure 2).

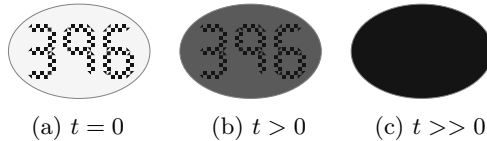


Fig. 2. Optical-scan oval with self-blanking confirmation code after being marked with an activator pen ($t = 0$ is the moment of activation)

Changes to the Protocols. The addition of self-blanking invisible-ink confirmation codes induces some changes the protocols presented in Section 3. Details are presented in the full paper² and are summarized as follows:

- **Ballot tuples:** \mathcal{P}_2 generates ballot-ids. Both printers run a *private printing* protocol to select a confirmation code and distribute it to VC shares,
- **Ballot printing:** \mathcal{P}_2 prints ballot-ids in invisible ink. Both printers print their shares of the confirmation codes using self-blanking visual crypto pixels,
- **Informational dispute resolution:** As in Scantegrity II, the printers only publish the confirmation code corresponding to the voted candidate. In the case of a dispute, the printers jointly issue a non-interactive proof of plaintext inequality between all remaining (unencrypted) codes on the disputed ballot.

5 Security Analysis of the Improved System

For space reasons we defer our security analysis to the full paper.² To briefly summarize our results, owing to the similarities between systems, we reduce the correctness of the improved system to that of Scantegrity II. Although Scantegrity has been peer reviewed and used in a real election we are not aware of a formal proof of the correctness. A proof of correctness of Eperio, a related

system, does offer some insight into how such a proof would proceed [16]. With respect to secrecy we present an argument that the improved system protects voter privacy even when one printer is corrupted. Assumptions regarding the physical primitives can be found there as well.

6 Related Work

We review some work related to verifiable voting systems with optical-scan paper ballots. This literature can be roughly separated into two categories: systems using single layer ballot forms but reliant on trusted parties/hardware and systems with distributed trust but with multi layer ballot forms.

Single Layer Ballot Forms with Trusted Components. The Scantegrity [9] and Scantegrity II [7,8,5] systems offer both a simple single layer fixed candidate list and an unencrypted paper trail, but make extensive use of trusted components to protect ballot secrecy including a computer for blackbox construction of the correctness proofs, the polling place scanner, the ballot printer as well as the custodian of cast ballots. Additionally the paper record reveals the link between receipt and clear-text vote making it unsuitable for public viewing. The Prêt-à-Voter [10,37] system and its variants [14,2,38] also offer the voter a single-layer ballot form with randomized candidate list. Although the correctness proofs are usually described as a multi-party computation, ballot forms are generated by a trusted printer. Cast ballots are generally “encrypted” though variants exist that leave a human readable paper trail [26,16]. Benaloh [2] proposes that receipts be generated and printed by a special-purpose device connected to the optical scanner. This has the distinct advantage that the ballots contain no identifying information (beyond the vote). However the issue of trusted ballot printing instead becomes a matter of trusted receipt printing.

Distributed Trust with Multi Layer Ballot Forms. Kubiak [22] and Carback et al. [19] propose *mostly* distributed modifications of the Punchscan system [35]. The former still relies on a trusted ballot printer, the later distributes printing but still relies on trusted hardware to generate ballot tuples. Carback and Popoveniuc [34] later propose a *three*-party distributed version of Punchscan in which top- middle- and bottom-sheet permutations are each generated by independent printing authorities. In all cases voters must use an indirect marking procedure. Moran and Naor [28] propose an improved multi layer ballot form that does not rely on indirection and with considerably stronger, provable, security properties. Voters are issued layers in separate sealed envelopes. Once inside the booth the voters are directed to remove each layers from its envelope and stack the layers in a particular order. The resultant candidate list is horizontally offset from the optical scan ovals by a randomized amount. Lundin et al. [25] propose a distributed construction of the Prêt-à-Voter ballot based on a form of dealerless 2-party visual cryptography. The voter must be careful to align the VC shares

in the booth in order to reconstruct the candidate list. Most recently Küsters et al. [23] present a version of Prêt-à-Voter system without a trusted printer, physically implementing a re-encryption mixnet using scratch-off coatings. The voter receives a separate ballot for *each* candidate, which can be cumbersome for races involving more than a few candidates.

Other Schemes. Chaum proposed the first physical receipt based voting system in [6]. It consists of two visual crypto layers showing the name of the voted candidate. A receipt is created by separating the layers and destroying one of them. Paul et al. [32] propose visual crypto for use in voter authentication for (non-cryptographic) remote voting systems. Scratch & Vote [1], Scratch, Click & Vote [24] and Pretty Good Democracy [38] make use of scratch-off coating to conceal encryption random factors and confirmation codes. Finally, Kelsey et al. [21] propose a voter-coercion strategy involving the use of scratch-off cards to direct voter action.

7 Future Work: Toward a Secure Multi-party Protocol

The systems described in this paper are both two-party protocols. Ultimately however it would be desirable to be able to distribute trust among arbitrarily many printers. With some modification the improved system presented in Section 4 could likely be extended to a secure multi-party protocol. With regard to creating the audit dataset this would be mostly a straightforward extension of the two-party approach with each of the $n > 2$ printers generating their own master permutations and issuing their own cut-and-choose proofs. Generating ballot tuples in a multi-party setting should also be a fairly straightforward extension of the two-party setting.

The primary challenge will be to develop an effective approach to distribute the ballot printing among more than two printers. This will undoubtedly require a fundamentally different approach from the two-party private printing scheme presented in [15] and is an interesting potential direction for future work.

Conclusion

Techniques for cryptographically verifiable elections offer unprecedented potential for making electronically-tabulated elections trustworthy. In this paper we presented two systems for cryptographically verifiable optical-scan voting that we believe offer properties that are desirable to both election officials and cryptographers. With this new approach election officials can continue to use a system with the familiar characteristics of optical-scan voting such as single layer ballots and paper audit trails which are both human-readable and conventionally auditable. Simultaneously the cryptographic audits can be conducted in a way that distributes trust such that no individual entity or piece of hardware has sufficient information to break voter privacy.

Acknowledgements. The authors wish to thank Jöern Mueller-Quade and the anonymous reviewers for their helpful feedback. Special thanks go to Jeremy Clark for many helpful discussions throughout the writing of this paper. This research is supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC)—the first author through a Postgraduate Scholarship and the third through a Discovery Grant. The second author was supported through a foreign exchange scholarship from the Karlsruhe House of Young Scientists (KHYS).

References

1. Adida, B., Rivest, R.L.: Scratch & vote: self-contained paper-based cryptographic voting. In: ACM WPES, pp. 29–40 (2006)
2. Benaloh, J.: Administrative and public verifiability: Can we have both? In: EVT (2008)
3. Benaloh, J.: Ballot casting assurance via voter-initiated poll station auditing. In: EVT (2007)
4. Benaloh (né Cohen), J.D., Fisher, M.J.: A robust and verifiable cryptographically secure election scheme. In: SFCS (1985)
5. Carback, R.T., Chaum, D., Clark, J., Conway, J., Essex, A., Hernson, P.S., Mayberry, T., Popoveniuc, S., Rivest, R.L., Shen, E., Sherman, A.T., Vora, P.L.: Scantegrity II election at takoma park. In: USENIX Security Symposium (2010)
6. Chaum, D.: Secret-ballot receipts: True voter-verifiable elections. *IEEE Security and Privacy* 2(1), 38–47 (2004)
7. Chaum, D., Carback, R., Clark, J., Essex, A., Popoveniuc, S., Rivest, R.L., Ryan, P.Y.A., Shen, E., Sherman, A.T.: Scantegrity II: end-to-end verifiability for optical scan election systems using invisible ink confirmation codes. In: EVT (2008)
8. Chaum, D., Carback, R., Clark, J., Essex, A., Popoveniuc, S., Rivest, R.L., Ryan, P.Y.A., Shen, E., Sherman, A.T., Vora, P.L.: Scantegrity ii: end-to-end verifiability by voters of optical scan elections through confirmation codes. *IEEE Transactions on Information Forensics and Security* 4(4), 611–627 (2009)
9. 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* 6(3), 40–46 (2008)
10. Chaum, D., Ryan, P.Y.A., Schneider, S.: A Practical Voter-Verifiable Election Scheme. In: De Capitani di Vimercati, S., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 118–139. Springer, Heidelberg (2005)
11. Clark, J., Hengartner, U.: On the use of financial data as a random beacon. In: EVT/WOTE (2010)
12. Clarkson, W., Weyrich, T., Finkelstein, A., Heninger, N., Alex Halderman, J., Felten, E.W.: Fingerprinting blank paper using commodity scanners. In: IEEE Symposium on Security and Privacy (2009)
13. 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)
14. Essex, A., Clark, J., Carback, R.T., Popoveniuc, S.: Punchscan in practice: an e2e election case study. In: WOTE (2007)
15. Essex, A., Clark, J., Hengartner, U., Adams, C.: How to print a secret. In: HotSec (2009)

16. Essex, A., Clark, J., Hengartner, U., Adams, C.: Eperio: Mitigating technical complexity in cryptographic election verification. In: EVT/WOTE (2010)
17. 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)
18. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: One-Time Programs. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 39–56. Springer, Heidelberg (2008)
19. Carback III, R.T., Popoveniuc, S., Sherman, A.T., Chaum, D.: Punchscan with independent ballot sheets: Simplifying ballot printing and distribution with independently selected ballot halves. In: WOTE (2007)
20. Jarrow, A., Pinkas, B.: Secure Hamming Distance Based Computation and Its Applications. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 107–124. Springer, Heidelberg (2009)
21. Kelsey, J., Regenscheid, A., Moran, T., Chaum, D.: Attacking Paper-Based E2E Voting Systems. In: Chaum, D., Jakobsson, M., Rivest, R.L., Ryan, P.Y.A., Benaloh, J., Kutyłowski, M., Adida, B. (eds.) Towards Trustworthy Elections. LNCS, vol. 6000, pp. 370–387. Springer, Heidelberg (2010)
22. Kubiak, P.: A modification of punchscan: Trust distribution. In: FEE (2006)
23. Küsters, R., Truderung, T., Vogt, A.: Improving and Simplifying a Variant of Prêt à Voter. In: Ryan, P.Y.A., Schoenmakers, B. (eds.) VOTE-ID 2009. LNCS, vol. 5767, pp. 37–53. Springer, Heidelberg (2009)
24. Kutyłowski, M., Zagórski, F.: Scratch, Click & Vote: E2E Voting over the Internet. In: Chaum, D., Jakobsson, M., Rivest, R.L., Ryan, P.Y.A., Benaloh, J., Kutyłowski, M., Adida, B. (eds.) Towards Trustworthy Elections. LNCS, vol. 6000, pp. 343–356. Springer, Heidelberg (2010)
25. Lundin, D., Treharne, H., Ryan, P.Y.A., Schneider, S., Heather, J., Xia, Z.: Tear and destroy: Chain voting and destruction problems shared by prêt à voter and punchscan and a solution using visual encryption. In: FEE (2006)
26. Lundin, D., Ryan, P.Y.A.: Human Readable Paper Verification of Prêt à Voter. In: Jajodia, S., Lopez, J. (eds.) ESORICS 2008. LNCS, vol. 5283, pp. 379–395. Springer, Heidelberg (2008)
27. Moran, T., Naor, M.: Basing Cryptographic Protocols on Tamper-Evident Seals. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 285–297. Springer, Heidelberg (2005)
28. Moran, T., Naor, M.: Split-ballot voting: Everlasting privacy with distributed trust. In: ACM CCS (2007)
29. Naor, M., Shamir, A.: Visual Cryptography. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 1–12. Springer, Heidelberg (1995)
30. Andrew Neff, C.: Practical high certainty intent verification for encrypted votes. Technical report, VoteHere Whitepaper (2004)
31. Park, C., Itoh, K., Kurosawa, K.: Efficient Anonymous Channel and All/Nothing Election Scheme. In: Hellese, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 248–259. Springer, Heidelberg (1994)
32. Paul, N., Evans, D., Rubin, A.D., Wallach, D.S.: Authentication for remote voting. In: HCSS (2003)
33. 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)
34. Popoveniuc, S., Carback, R.: Clearvote: An end-to-end voting system that distributes privacy between printers. In: WPES (2010)

35. Popoveniuc, S., Hosp, B.: An introduction to punchscan. In: WOTE (2006)
36. Rabin, M.: Transaction protection by beacons. *Journal of Computer and System Sciences* 27(2) (1983)
37. 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)
38. Ryan, P.A., Teague, V.: Ballot permutations in prêt à voter. In: *EVT/WOTE (2009)*
39. Ryan, P.A., Teague, V.: Pretty good democracy. In: *Workshop on Security Protocols (2009)*
40. Sandler, D.R., Derr, K., Wallach, D.S.: VoteBox: a tamper-evident, verifiable electronic voting system. In: *USENIX Security Symposium (2008)*
41. Sherman, A.T., Carback, R.T., Chaum, D., Clark, J., Essex, A., Hernson, P.S., Mayberry, T., Popoveniuc, S., Rivest, R.L., Shen, E., Sinha, B., Vora, P.L.: Scantegrity mock election at takoma park. In: *EVOTE (2010)*
42. Xia, Z., Schneider, S.A., Heather, J.: Analysis, improvement and simplification of prêt à voter with paillier encryption. In: *EVT (2008)*

Paperless Independently-Verifiable Voting*

David Chaum, Alex Florescu¹, Mridul Nandi³, Stefan Popoveniuc, Jan Rubio¹,
Poorvi L. Vora^{1,2}, and Filip Zagórski¹

¹ The George Washington University, Washington D.C.

² Indian Institute of Technology, Bombay

³ Indian Statistical Institute, Kolkata

Abstract. We present a new model for polling-booth voting: the voter enters the polling booth with a computational assistant which helps her verify that her vote is correctly recorded. The assistant interacts with the voting system while the voter votes on the machine in the polling booth. We present an independently-verifiable, coercion-resistant protocol based on this model. Unlike all other independently-verifiable protocols, this one is completely paperless and does not require the voter to perform any tasks outside the polling booth. We provide property definitions, rigorous claims and a description of a prototype.

1 Introduction

Independently-verifiable protocols were first proposed almost a decade ago, and have been tried in binding elections, including one small governmental election. All of the secure independently-verifiable protocols require the use of paper, however, and also require the voter to perform checks outside the polling booth. This has perhaps slowed down the adoption of these protocols, which have particularly strong verifiability properties. Additionally, Direct Recording Electronic (DRE) voting machines—with all their flaws—enable voters with disabilities to vote independently for the first time ever. The use of paper hence presents a step backwards for this category of voter. In a first attempt towards rectifying some of these problems, this paper presents an independently-verifiable polling-booth protocol which is completely paperless.

The protocol is based on a new model for independently-verifiable polling booth voting: the voter enters the polling booth with a computational assistant that she has brought with her. This could be, say, a smartphone, or special-purpose hardware. She puts it into a special docking station where she can see its output behind a transparent cover (much like the screen used for VVPATs), but cannot provide any input to it. A blind voter may obtain output through the use of headphones, and be prevented from providing input by disabling the microphone; again, specially-designed hardware could be useful. We assume that the voter takes no pictures and votes alone so no one can watch her vote. An adversary may, however, query her through alternate channels (such as the scratch-off cards proposed in [13]).

* This work was done while Nandi was at The George Washington University. Florescu, Rubio and Vora were supported in part by NSF Award No. 0831149, Nandi and Zagórski by NSF Award No. 0937267. Zagórski was also supported in part by the Polish Ministry of Science and Higher Education scientific project - grant N N206 369839.

In the protocol, while the voter votes on the machine in the polling booth (this could be a DRE machine that is capable of performing cryptographic operations), the assistant interacts with the machine. It performs computations based on the interaction, and provides information to the voter to help her determine if her vote is recorded correctly. The role of the assistant is to provide the voter the capability to perform digital signatures and commitments, and to perform the random challenges on behalf of the voter. The assistant cannot determine the vote from the information it obtains. The data communicated in both directions between assistant and voting machine is signed by the sender and verified by the receiver, and made public immediately after the protocol ends. Hence, as with other independently-verifiable protocols, the encryption-correctness proof—which is based on the interaction between the computational assistant and the voting machine—can be checked by anyone. In contrast with existing independently-verifiable protocols, the voter’s tasks are completed inside the polling booth, and the process is entirely paperless. We do not assume the existence of a randomness beacon.

We present a coercion-resistant independently-verifiable fully-electronic protocol using this model. We are not aware of any other such protocols that do not require an external independent randomness beacon.

1.1 Existing Paperless Protocols and Their Limitations

It may be argued that the design of an independently-verifiable paperless protocol has been the goal of cryptographers since the invention of secure electronic voting in the early eighties. However, all the existing all-electronic protocols have vulnerabilities.

The classical protocols are vulnerable because the vote is entered through the voter’s machine which is trusted to keep the vote secret. Malware on the voter’s machine knows the vote, and can also change it.

The newer, all-electronic independently-verifiable voting systems may be represented by *polling-booth-Helios*. While Helios is a remote voting system, it is an all-electronic simplification of polling-booth protocols Simple-Verifiable-Voting [3] or Voter-Initiated Poll Station Auditing [4] which provide paper receipts. It is hence easily modified for polling booth use, and we use it for illustrative purposes. When we refer to polling-booth-Helios in this paper, we will mean an all-electronic polling-booth protocol where the voter communicates an electronic vote to the untrusted voting machine, obtains an electronic hash of her receipt as a commitment from the machine, communicates electronically whether she wants to cast (or audit) the receipt, and receives an electronic receipt (or proof of encryption-correctness). We use it to represent a simple all-electronic independently-verifiable polling-booth system.

Polling-booth-Helios is vulnerable to two types of attacks:

1. Coercive attack: the adversary coerces the voter to make a challenge that is a function of the receipt hash. Because the voter does not know whether her vote will be cast or audit when she is entering it, and she knows that there is a possibility it will be audited, or exposed to the adversary, she has an incentive to vote as directed by the adversary. Further, if this adversary colludes with the voting system, the latter’s attempts at cheating cannot be detected because it will know what challenge to expect. This attack can be avoided if the voter is required to commit to her challenge

bit before she sees any information on the ballot, however a voter cannot commit to the bit on an all-electronic system without access to trusted computation¹.

2. Challenge correctness: the voting system can choose to ignore the challenge and proceed to cast or audit as it wishes. While the voter knows the system is cheating, she cannot prove it. The inability to resolve a dispute about the correctness of the challenge can also allow a voter to falsely claim a voting system is cheating.

1.2 Our Contributions

We address the problem of making and checking commitments in an all-electronic protocol by having the computational assistant perform a more active role than in the current independently-verifiable voting systems. The assistant makes and verifies cryptographic commitments and digital signatures on behalf of the voter. We avoid the coercion attacks possible in the classical cryptographic protocols—where too the voter’s computer performed an active role—by preventing the voter from providing any input to the assistant. Finally, we make fewer assumptions than in the classical model where the voter’s computer is trusted to follow instructions. We also do not assume the availability of a beacon of randomness. Our contributions are as follow:

- ◊ We model an *Actively-Assisted-Human Interactive Proof (AAHIP)*. In this model the voter votes on the voting machine and does not provide any input to the assistant. The assistant interacts with the voting machine and provides information to the voter. The interaction between assistant and voting machine takes place over an authenticated channel, where both assistant and voting machine sign messages and check signatures. All the data sent over the channel, in both directions, is made public immediately after the protocol ends. The assistant is allowed to deviate from protocol, and its memory and logs can be examined by anyone once the vote is cast. We also allow the assistant to instruct the voter before the protocol begins, and to query her at any time.
- ◊ We provide a rigorous description of a *paperless* vote-casting protocol based on a cut-and-choose AAHIP. We provide definitions and rigorous statements of our results. Space limitations prevent us from providing proofs. We assume the model of an AAHIP and that either the voting machine or the assistant is honest. We define *soundness* and *coercion-resistance* and demonstrate that the protocol achieves both given the assumptions. We are not aware of other all-electronic protocols that have these properties. In particular, note that when the assistant is dishonest the protocol is still coercion-resistant. Classical all-electronic protocols are coercion-resistant only when the voter’s computer is trusted.

¹ Paper-based systems can enable the voter to make a commitment without using computation—for example, poll workers may mark the voter’s choice of challenge on her paper ballot before handing it to her. In an all-electronic voting system, however, without access to any other computation while voting, the voter is unable to make the commitment. In order to do so, the commitment would need to be digital and the voter would need access to a trusted computer. This computer could change the challenge as it wished without informing the voter. An audit challenge would result in an audit of both the voter and the voting machine. In trying to avoid one coercive adversary, the Helios voter is exposed to another.

- ◇ We describe a simpler challenge-response **AAHIP** for vote-casting, which is much like polling-booth-Helios with an assistant. We show that it has weaker coercion-resistance. Thus the vulnerabilities of polling-booth-Helios are not overcome by naturally extending it to the **AAHIP** model. We are not aware of any other work that observes a distinction between cut-and-choose and challenge-response protocols.
- ◇ We describe our prototypes for both challenge-response and cut-and-choose protocols. An Android-based smartphone performs the role of the computational assistant in the prototypes. Note, of course, that the voter's device can be a special hardware device and not a smartphone; at this time, however, we do not have access to special hardware.

1.3 Comparison with Other Approaches

Our protocol provides the following security improvements over polling-booth-Helios (note that the remote version also possesses these vulnerabilities, but they may not be of as much consequence in the elections Helios was designed for). Again, polling-booth-Helios is being used only for illustrative purposes, as a representative simple all-electronic system:

- ◇ In contrast with polling-booth-Helios, the voter can prove that a challenge bit was changed if the assistant is honest.
- ◇ Further, in contrast with polling-booth-Helios, the voter cannot be coerced if the voting machine is honest.

We are able to achieve the above two properties because we have constructed a protocol where the challenge bit is issued by the assistant and not by the voter. One cannot simply add digital signatures to the challenge bit in Helios to achieve the first property, because the voter cannot check digital signatures without an assistant. Further, the second property is not achieved by naturally extending Helios to use a computational assistant to issue the challenge bit and to make and verify digital signatures. We show that a protocol that is a natural extension of Helios with a computational assistant is not coercion resistant, even if it is assumed that the voter does not provide input to the assistant. That is, it is not possible to achieve the properties achieved by our main protocol by simply extending Helios to the **AAHIP** model.

On the other hand, our protocol also shares some of the limitations of polling-booth-Helios and does not achieve all the security properties of paper-ballot-based voting systems. If the voting machine is dishonest, an honest assistant can help the voter detect the cheating, but neither can prove it. In a paper-ballot-based system, the voter and one of the computational assistants she uses can provide the proof.

Note that one may fault our protocol because, if the assistant and voting machine are both dishonest and colluding, the voter will not detect an attempt to change the tally. While this is possibly a consequence of more general results on protocols in which more than half of the parties are dishonest, note that polling-booth-Helios also has a similar problem. Consider an adversary colluding with the voting system to change the tally in our protocol. The adversary prevents the voter from using her chosen assistant and requires her to use one the adversary provides. This assistant can provide pre-prepared

challenges known to the voting system. This will result in a fraudulent encryption-correctness proof. The same adversary can use the coercion channel present in polling-booth-Helios to force the polling-booth-Helios voter to execute a challenge that is a function of the receipt-hash. In collusion with this adversary, the polling-booth-Helios voting system can also predict challenges, also resulting in a fraudulent encryption-correctness proof. However, in our protocol, the coercive adversary cannot coerce the voter if the voting machine is honest; this is not true with polling-booth-Helios.

1.4 Organization

The paper is organized as follows. Section 2 reviews related work. Section 3 provides an informal description of the model. Section 4 describes the challenge-response protocol which is a natural extension of polling-booth-Helios to the AAHIP model, and describes a simple coercion attack on it. The main protocol is described informally in 5. Section 6 provides a rigorous description of the model and the main protocol. Section 7 provides a rigorous set of property definitions with theorem statements, and section 8 describes the prototypes. Section 9 presents our conclusions.

2 Related Work

The first description of the notion of coercion-resistance is due to Benaloh and Tuinstra [5]. Their protocol assumes the existence of a randomness beacon, avoiding the coercion avenue created when voters are allowed to issue the challenges. Later protocols in the classical model, such as that of Juels, Catalano and Jakobsson [11] assume that the computer used to encrypt the vote may be trusted to do so correctly (or that voters are Interactive Turing Machines — ITMs). This is clearly not a valid model for polling-place voting. The first description of a secure voting protocol where voters are not ITMs is due to Chaum [7]; however this protocol requires the use of visual cryptography and two transparent layers stacked one on top of the other with very good registration and is highly impractical. These protocols were quickly followed by several others, such as MarkPledge [14], Prêt à Voter [16], Punchscan [15], Simple-Voter-Verifiable [3], Votebox [17], Helios [2] and Scantegrity [8].

Of the independently-verifiable protocols, most are paper-based. Votebox and Helios, based on Simple-Voter-Verifiable, are fully-electronic, but limit themselves to elections where coercion is not a concern.

The protocols we describe and study in this paper were first informally proposed by Chaum, Popoveniuc and Vora [9]. We present slightly modified versions in this paper. The original paper does not contain rigorous descriptions or proofs and they do not present any information on prototypes. Additionally, the original paper was not presented at a venue with proceedings.

Our notion of an AAHIP extends Adida’s work on AHIPs [1].

3 The Informal Model

In our protocols, we use the notion of a commitment, and that of oblivious transfer. A cryptographic *commitment scheme* enables a sender to “commit” to a value M without

revealing it. It does so by providing a “commitment”, com_M , to the receiver. At a later stage, the sender can “open” the commitment and reveal M to the receiver. The receiver can verify that M is the original value committed to. *1-out-of-2 oblivious transfer* is a protocol between a sender and a receiver by which the receiver can obtain only one of two secret elements K_0 and K_1 from the sender. The sender is oblivious of which of the two values the receiver received.

In all the protocols described here, there are three participants: voter, voting machine and computational assistant. The voter is human. Its computational capability is limited to the ability to compare small strings. The voting machine and computational assistant are Probabilistic Polynomial Time (PPT) Interactive Turing Machines (ITMs) with authenticated write access to a secure append-only bulletin board that can be read by anyone.

The general purpose of all the participants is as in other independently-verifiable voting systems, except for two important distinctions: the computational assistant is an interactive participant in the protocol and does not receive any input from the voter. All its input is provided by the voting machine.

We provide more detail on the similarities. As before, the voting machine presents the voter with a ballot. The voter votes on the voting machine, which provides a string which it claims is the encryption of the vote. The voting machine provides an interactive proof supporting this claim. The computational assistant checks the correctness of the proof transcript without knowing how the voter voted. The voter performs a check inside the booth. The voting machine is said to provide a correct encryption only when both checks are passed.

Details on the distinctions are as follow.

- ◇ The computational assistant (and not the voter) obtains the vote encryption and provides challenges to the voting machine to obtain a proof that the encryption is correct.
- ◇ The assistant provides information to the voter based on how the voting system responds to the challenges. This information may consist of more than the binary outcome of its checks. It may also include information on what should be on the ballot (for example, the correct ordering of candidates, or the correct association of candidates with dummy variables). The voter compares this information to that on the ballot presented to her by the voting machine. If the information matches, she accepts that the encryption is correct.
- ◇ The voter does not provide any information to the assistant and only receives information from it.
- ◇ The voter has to choose a single assistant to participate in the protocol (in an AHIP, the voter can present her receipt to several distinct assistants asking each to check for her).

All interaction between the two ITMs is signed and verified. If a signature does not verify, the recipient party requests a resend and aborts after a pre-determined fixed number of failed attempts to verify. Similarly, when a commitment is not opened correctly, the aggrieved ITM can abort the protocol. Thus, either party can perform a denial-of-service. As this possibility exists in the use of all computational devices, we do not consider it any further. Note that, because information between the two ITMs is signed

and published after the protocol ends, the channel between the two parties is much like a public append-only channel. The data across this channel will demonstrate which party has cheated.

The voter can also abort the protocol if she catches a party behaving dishonestly. However, because he is not an ITM, she cannot sign and verify signatures and hence does not share a verifiable tape with any party. Hence she is typically not able to prove certain types of dishonest behavior.

The assistant would typically be provided by an individual or organization the voter chose. This is not to say the individual or organization is not malicious or dishonest and will not attempt to coerce the voter to vote in a certain manner. Our main protocol is coercion-resistant if at least one of the voting machine or assistant is honest and the voter does not provide any input to the assistant during the protocol. This is true even if the adversary can query the voter and examine the memory and logs of the assistant after the vote is cast and if the assistant is allowed to deviate from protocol.

4 The Challenge-Response Protocol and a Weakness

In this section we present a slight generalization of the protocol named ϵ Tegrity in [9]—a challenge-response-style protocol which is the natural extension of Helios to the AAHIP model. The voter familiar with the ballot audits of Prêt à Voter [16] or Scantegrity [6] will notice the similarity with ballot audits. We name the generalized protocol ϵ ChallengeResponse. The protocol uses the assistant to make and verify commitments and digital signatures in a straightforward manner. Its simplicity makes it easy to use. Space restrictions prevent us from going into more detail. We describe a coercive attack on this protocol, illustrating the difficulty of designing a coercion-resistant all-electronic protocol.

In this protocol, the assistant first commits to the challenge bit on behalf of the voter. It sends the challenge bit to the voter, and the commitment to the voting machine. The voter enters her vote and the voting machine sends the vote-encryption to the assistant. The assistant opens its commitment and the voting machine audits or casts the encryption based on what the challenge bit is. The assistant checks encryption-correctness if the challenge corresponds to an audit. After describing the protocol we describe a simple coercive attack, thus motivating the somewhat more complicated protocol which is our main contribution. The challenge-response protocol and the coercive attack are interesting because they illustrate the subtle problems involved in designing an incoercible protocol.

4.1 ϵ ChallengeResponse: Informal Description

In general, an independently-verifiable voting system will provide a receipt which functions as an encryption of the vote. We will denote by \mathcal{E}_s the cipher for the ballot with serial number s . That is, whether \mathcal{E}_s is implemented as a cipher or a look-up table, it has the properties of a secure cipher. In order to vote, a voter walks into a polling booth with a computational assistant.

To begin the protocol, the assistant and the voting machine set up a communication link. Before the voter and voting machine interact, the assistant performs a cryptographic commitment to a uniformly-distributed bit b representing whether or not it will audit the upcoming encryption. It sends b to the human voter and the commitment c_b to the voting machine. Note that it has not been possible to perform this step electronically in an AHIP, and independently-verifiable voting systems have required a combination of paper and procedures for this step [2](#).

The voting machine sends to the assistant the serial number s of the ballot it will use. After the assistant checks that s is fresh, the voting machine presents the fresh ballot with serial number s to the voter. The voter selects her candidate, v , on the voting machine and confirms her vote. The voting machine then presents the signed pair of the serial number and encrypted vote, $(serial, encryption)$, to the assistant. In correct instances of the protocol, $(serial, encryption) := (s, \mathcal{E}_s(v))$.

The assistant opens c_b to reveal b to the voting machine.

- ◊ If the commitment verifies and corresponds to a choice of “cast”, the vote is cast. Both voting machine and assistant post the signed pair $(serial, encryption)$ in the list of verified votes on the secure bulletin board. The protocol ends.
- ◊ If the commitment verifies and corresponds to a choice of “audit”, the voting machine reveals the vote it encrypted, $vote$, and the encryption parameters. Both voting machine and assistant post the signed pair $(serial, encryption)$ in the list of audited votes on the secure bulletin board. The assistant provides the value of $serial$ and $vote$ to the voter. If $vote$ is the vote she cast, i.e. if $vote = v$, and $serial$ is the serial number of the ballot presented to her, i.e. if $s = serial$, the voter knows the encryption was correctly performed.

The voter may repeat the above protocol many times if she desires, at different times during the day, and at different voting machines.

4.2 A Coercive Attack

We now describe a simple coercion attack when the voting machine is honest but the assistant is not. This motivates the use of the more complicated cut-and-choose protocol we describe next, which is not vulnerable to this attack. The coercer tells the voter to vote for candidate i and provides her with the assistant. An examination of the assistant after the vote is cast will demonstrate to the coercer whether the voter used it or not, so the voter cannot use another assistant. The assistant is programmed to perform the correct protocol, except for the fact that it provides no information to the voter in the first step (when it is supposed to send her the challenge bit). The voter will always

² As in an AHIP, the voter may choose b and communicate it to the assistant. However, because the assistant commits to b , any security analysis must assume that the assistant chooses b . Further, the coercion-resistance analysis also requires that the voter not be allowed to provide input to the assistant, so, if the voter were allowed to choose this bit, she should not be allowed to provide any other information to the assistant. This bit itself does not provide any information because it could as well have been communicated outside the booth. (It is obtained from the voter before she herself has been provided any information in the polling booth).

vote for candidate i because she does not know when *her* commitment to her vote (the encrypted vote) will be audited by the coercer (assistant).

We now propose the following, more complicated protocol, based on a cut-and-choose approach which we implement with the use of 1-out-of-2 Oblivious Transfer.

5 The Cut-and-Choose Protocol

In this section we informally describe the protocol named ePunchscan in [9] with a few modifications; our protocol is named eCutAndChoose. It is a cut-and-choose protocol for a commitment-based cryptographic voting system. It uses a mix of ideas from Prêt à Voter, Punchscan and Scantegrity. Its most striking aspect, the use of oblivious transfer, is, however, common only to Punchscan among the independently-verifiable protocols that have been used in real elections.

The ballot consists of a serial number s and two ballot parts: (i) $Part_0$ contains a permutation π_s of the c candidates, reflecting the order in which candidates will be presented to the voter (ii) $Part_1$ consists of a list of Scantegrity codes, one for each of c candidate positions. Each part bears the serial number s . $Part_0$ also bears another serial number, s_L .

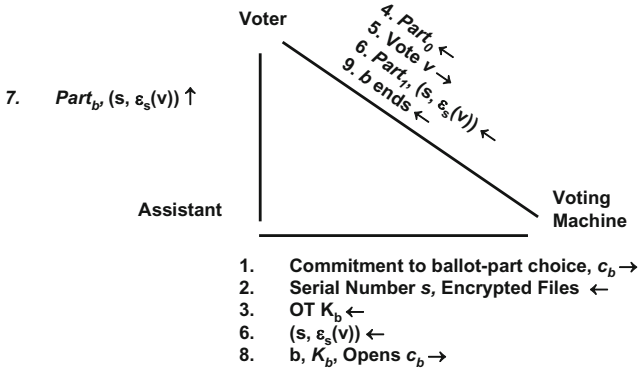


Fig. 1. The Cut-And-Choose Protocol. Arrows denote direction of communication.

5.1 Protocol Description

Before the election starts, the voting system makes cryptographic commitments to the two ballot parts separately, as well as to the commitments required by a Scantegrity back-end that will obtain vote tallies from confirmation codes. To begin, a voter walks into a polling booth with a computational assistant. The assistant and the voting machine set up a communication link.

Commitment to Choice Of Ballot Part: [Step 1 in Figure 1] The assistant performs a cryptographic commitment to a bit b representing the ballot part it will choose to get by

Oblivious Transfer. It sends b to the human voter and the commitment c_b to the voting machine. Let us say the voter gets bit . In correct instances of the protocol, $bit := b$. Note that, in AHIP-based Punchscan, this step is performed using a combination of paper and polling procedures. Polling officials note down whether a voter will take home the top or bottom layer before the voter is allowed to see the ballot³.

Ballot Preparation: [Step 2] The voting machine sends to the assistant the serial number s of the ballot it will use, and the assistant checks that s is fresh. The voting machine creates two ciphertexts. The i^{th} ciphertext, $i = 0, 1$ is a symmetric-key encryption of $File_i$, which consists of $Part_i$ and the information required to verify its published commitments. $File_i$ is encrypted with symmetric-key encryption using key K_i for $i = 0, 1$. Each key is pseudo-randomly generated afresh for each ballot.

Oblivious Transfer: [Step 3] The assistant obtains K_b from the voting machine by oblivious transfer, obtains $File_b$ by decrypting the appropriate ciphertext, and checks the commitments to $Part_i$.

	35967		35967		35967
Bob	293	Bob		Alice	447
Alice	447	Alice		Carol	530
Carol	530	Carol		Confirmation Code	447
Confirmation Code	447	Confirmation Code	447	Confirmation Code	447

Displayed on Voting Machine Displayed on Assistant if Assistant obtains Ballot Part₀ Displayed on Assistant if Assistant obtains Ballot Part₁

Fig. 2. Voter's View of the Ballot After Casting a Vote for Alice

Ballot Presentation: [Step 4] The voting machine presents $Part_0$ of the ballot with serial number s to the voter. This looks much like a Prêt à Voter ballot.

Voting: [Steps 5] The voter enters her vote v on the voting machine and confirms it.

Receipt: [Step 6] The voting machine now presents $Part_1$ and the traditional Scantegrity confirmation number, $\mathcal{E}_s(v)$, to the voter. It also presents the signed confirmation number to the assistant.

Receipt: [Step 7] The assistant presents $Part_b$ and the confirmation code to the voter. The voter checks that the ballot part and confirmation number presented by the assistant matches the corresponding part and confirmation number presented by the voting machine. (See Figure 2)

Closing: [Steps 8-9] The assistant opens its commitment to bit b for the Voting Machine and also provides K_b . The Voting Machine provides bit b to the voter, so it may know how to lie about the ballot half the assistant does not have if coerced by the assistant. The voting machine casts the vote by posting the signed pair of serial number and vote

³ As in an AHIP, the voter may choose b and communicate it to the assistant. However, because it is the assistant that performs the cryptographic commitment to b , any security analysis must assume that the assistant chooses b . Further, as in eChallengeResponse, the voter must not be allowed to provide any other information to the assistant during the protocol.

encryption in the list of cast votes on the secure bulletin board. It informs the voter and assistant that the vote is cast. The assistant posts the signed pair of serial number and vote encryption it received from the voting machine in Step 6 in the list of verified votes on the secure bulletin board. The protocol ends.

5.2 Dispute Resolution

Notice that, if the voter and the voting machine disagree about what transpired between them over their private channel, it is not possible to resolve the dispute except through physical observation. So, for example, the voting machine might constantly behave as though the voter voted $v' \neq v$, or might refuse to abort when instructed to, etc. While the voter would be aware that the voting machine was cheating, she would not be able to prove it as the tape between the two parties is not verifiable and can be easily overwritten by the voting machine. While this is not as bad as the original problem with Direct Recording Electronic voting machines, it demonstrates the difference between the use of electronic and paper tapes (paper ballots) between voting machines and voters. Every interaction between voter and voting system in paper ballot independently-verifiable systems such as Prêt à Voter and Scantegrity is on an append-only write-once tape such as a paper ballot or through physical processes. This allows the voter to provide evidence when the voting system does not follow protocol. Notice that this can be a problem even if the voting system is honest, when the voter is dishonest. A small group of vocal dishonest voters can call into question an honest election by accusing the voting machine of not following protocol.

A possible solution to this problem is to allow the voter to interact with the assistant to obtain a blind signature on her confirmation code and to vote by casting the blinded confirmation code, through the assistant. The voting machine then will not know what the vote is and cannot change it. It is not clear whether this opens the voter up to more coercion-attacks, and whether there are verifiability and security problems with this protocol that are similar to those found in the past with blind-signature-based protocols.

6 Rigorous Description: eCutAndChoose

In this section, we provide a rigorous description of the model and of the informally-described cut-and-choose protocol in [9], where it is referred to as ePunchscan. We describe only set-up and vote-casting; vote tallying proceeds as with Scantegrity.

6.1 The AAHIP Model

We do not describe the most general version of this model here. We focus only on the specific case when a voting machine seeks to prove to a voter that a string x is an encryption of her vote v using a cipher \mathcal{E} . There are three participants in the protocol: the voter \mathcal{V} ; the voting machine \mathcal{VM} and the assistant \mathcal{A} . \mathcal{V} is a human whose computational capability is limited to the comparison of strings of size n bits. \mathcal{VM} and \mathcal{A} are PPT ITMs. There are the following channels among the participants:

1. A two-way private channel between \mathcal{V} and \mathcal{VM} provided by the poll booth. No one other than \mathcal{V} or \mathcal{VM} can read from, or write to, this channel.
2. A two-way authenticated channel between \mathcal{A} and \mathcal{VM} made append-only through the use of digital signatures. This channel can be read by anyone at any time. Only \mathcal{A} and \mathcal{VM} have read-write access to it.
3. A one-way channel from \mathcal{A} to \mathcal{V} which is private during the voting process and public thereafter. Only \mathcal{A} can write to it.

Note that there is no channel from \mathcal{V} to \mathcal{A} during the protocol; however, \mathcal{A} may query \mathcal{V} after the vote is cast.

The view of participant \mathcal{X} is denoted $view_{\mathcal{X}}$. The transcript of the interaction between participants \mathcal{X} and \mathcal{Y} is denoted $\text{Tape}_{\mathcal{X},\mathcal{Y}}$.

The protocol takes as input (private) vote v from \mathcal{V} and (public) challenge bit b from \mathcal{A} . We denote it $\mathcal{AAHIP}(\mathcal{V}(v), \mathcal{A}(b))$. It produces the following output:

1. The receipt, public output from \mathcal{VM} :

$$(\text{Receipt}_s, \text{Receipt}_E, \text{Receipt}_P) :=$$

$$\text{Receipt}(v, b) = (s, \mathcal{E}_s(v), \text{Proof}(\mathcal{E}_s(v), b))$$

where s is the serial number of the ballot, $\mathcal{E}_s(v)$ the claimed encryption of the vote, and $\text{Proof}(\mathcal{E}_s(v), b)$ the proof that $\mathcal{E}_s(v)$ is correctly constructed, for challenge b . It consists of the transcript between \mathcal{VM} and \mathcal{A} .

2. The check of the assistant, public output (True or False) from \mathcal{A} :

$$\text{CheckProof}(\text{Receipt}(v, b))$$

indicating whether the commitments are correctly opened by \mathcal{VM} in Receipt . This is similar to the output produced by \mathcal{A} in an AHIP .

3. The ballot-part, (private during the protocol but public after the vote is cast) from \mathcal{A} to \mathcal{V} :

$$\text{Part}(view_{\mathcal{A}})$$

which represents what \mathcal{A} has learnt about the manner in which $\mathcal{E}_s(v)$ was constructed by \mathcal{VM} . In our cut-and-choose protocol, $\text{Part}(view_{\mathcal{A}})$ is the ballot part obtained by oblivious transfer.

4. The check of the voter, public output (True or False) from \mathcal{V} :

$$\text{CheckPart}(\text{Part}(view_{\mathcal{A}}), view_{\mathcal{V}})$$

which indicates whether $\text{Part}(view_{\mathcal{A}})$ is consistent with what \mathcal{V} observes in the private channel with \mathcal{VM} . In our cut-and-choose protocol, CheckPart indicates whether the chosen part—as determined by the assistant through the opening of the original commitments—matches the corresponding part in the ballot displayed to the voter.

6.2 Initial Set-Up by Election Officials

Let \mathcal{C} define the Scantegrity code-space; that is, it is the set of all possible Scantegrity confirmation codes. It can be any set such that (a) individual elements of the set can be compared by humans (that is, each code is no longer than n bits long) and (b) the size of \mathcal{C} is large enough so that probability $\epsilon := |\mathcal{C}|^{-1}$ is small enough.

We consider each race separately. Let c be the number of candidates in the race and N the number of ballots to be generated. Election officials (EOs) perform the following tasks prior to the election:

1. Generate confirmation codes $\text{code}_{s,i}$, for all serial numbers $1 \leq s \leq N$ and candidates $i \in \mathbb{Z}_c$. The c confirmation codes on a single ballot are distinct. That is,

$$\text{code}_{s,i} \neq \text{code}_{s,j} \quad i \neq j, \quad i, j, \in \mathbb{Z}_c \quad 1 \leq s \leq N$$

Informally speaking, $\text{code}_{s,i}$ should be the secure symmetric-key encryption of i corresponding to ballot s . That is, the function $f : \{0, 1, 2, \dots, N\} \times \mathbb{Z}_c \rightarrow \mathcal{C}$ with $f(s, i) = \text{code}_{s,i}$ should have the properties of a symmetric-key encryption (with keyspace $\{0, 1, 2, \dots, N\}$, message space \mathbb{Z}_c and ciphertext space \mathcal{C}) where the key is a secret function of s . The scheme should be such that the advantage of a PPT adversary in the eavesdropping indistinguishability experiment [12, page 63] is negligible in the length of s .

2. Generate a secret pseudo-random permutation $\pi_s(\cdot)$ of \mathbb{Z}_c (the candidates) which is independent of $f(s, \cdot) \forall s$.
3. Generate the ballots, as pairs. For each serial number s , generate the ballot: $(s, \text{Part}_0, \text{Part}_1)$ where $\text{Part}_0 = \pi_s$ and $\text{Part}_1 = \pi_s(\langle \text{code}_{s,i} \rangle_{i \in \mathbb{Z}_c})$.
4. Generate commitments to the correspondence between candidate and confirmation code for each ballot and candidate separately, as well as commitments to both parts separately for each ballot using commitment scheme \mathfrak{C} (see Appendix for definition)

$$\begin{aligned} (\text{com}_{s,i}, \text{open}_{s,i}) &= \mathfrak{C}_{\text{EO}}(s, \text{code}_{s,i}) \quad \forall s, i \in \mathbb{Z}_c \\ (\text{com}_{s,\text{part}_0}, \text{open}_{s,\text{part}_0}) &= \mathfrak{C}_{\text{EO}}(s, \text{part}_0) \quad \forall s \end{aligned}$$

and

$$(\text{com}_{s,\text{part}_0}, \text{open}_{s,\text{part}_0}) = \mathfrak{C}_{\text{EO}}(s, \text{part}_0) \quad \forall s$$

5. Keep a secret record of

$$\begin{aligned} &(\text{open}_{s,\text{part}_0}, \text{open}_{s,\text{part}_1}, \text{open}_{s,0}, \text{open}_{s,1}, \dots \\ &\dots, \text{open}_{s,i}, \dots, \text{open}_{s,c-1}) \quad \forall s \end{aligned}$$

and publish the values of

$$\begin{aligned} &(\text{com}_{s,\text{part}_0}, \text{com}_{s,\text{part}_1}, \text{com}_{s,0}, \text{com}_{s,1}, \dots \\ &\dots, \text{com}_{s,i}, \dots, \text{com}_{s,c-1}) \quad \forall s \end{aligned}$$

on a secure public bulletin board.

6.3 Casting a Vote

As with `eChallengeResponse`, this protocol is interactive among three parties: voter \mathcal{V} , voting machine \mathcal{VM} and assistant \mathcal{A} . The steps below are numbered as in Figure 11. Note that any time a party finds that a commitment is not opened correctly, or finds that a signature is not verified, or notices that another party fails a check, it aborts the protocol.

1. \mathcal{A} chooses $b \in \{0, 1\}$ ($b = i$ implies that \mathcal{A} will check $Part_i$). \mathcal{A} sends signed commitment c_b to \mathcal{VM} where $(c_b, o_b) \leftarrow \mathfrak{C}_{\mathcal{A}}(b||\bar{b})$.
2. \mathcal{VM} verifies signature on c_b . \mathcal{VM} chooses a fresh serial number s at random and generates two secret encryption-keys K_0, K_1 at random. \mathcal{VM} sends a signed message to \mathcal{A} consisting of the serial number s and the ciphertext of two files $C_i = E(K_i, File_i)$, $i = 0, 1$, where $File_i = (Part_i||open_{s,part_i})$. E denotes a secure symmetric-key encryption scheme.
3. \mathcal{A} verifies the signature and checks on the secure bulletin board that s is unused. \mathcal{A} performs an oblivious transfer with \mathcal{VM} to obtain K_b : $OT_{\mathcal{A},\mathcal{VM}}(b, K_0, K_1)$. \mathcal{A} decrypts $File_b$ with K_b and verifies the opened commitments in $File_b$. \mathcal{A} sends the “ready” signal to \mathcal{V} .
4. \mathcal{VM} presents $File_0$ with serial number s to \mathcal{V} .
5. \mathcal{V} sends vote v to \mathcal{VM} .
6. \mathcal{VM} sends the signed pair $(serial, code) := (s, code_{s,v})$ —signed serial number and confirmation code—to \mathcal{A} and to \mathcal{V} . It also sends $Part_1$ to \mathcal{V} .
7. \mathcal{A} verifies the signature and that $serial = s$. \mathcal{A} presents $Part_b$ and the pair $(serial, code)$ to \mathcal{V} . \mathcal{V} checks that $Part_b$ as presented by \mathcal{VM} is identical to $Part_b$ as presented by \mathcal{A} , and that the confirmation and serial numbers presented by both match. If they match, \mathcal{V} sets $CPart := \text{CheckPart}(\text{Part}(view_{\mathcal{A}}), view_{\mathcal{V}}) = \text{“True”}$. Practically speaking, this is conveyed when \mathcal{V} leaves the polling site without complaint. We denote the two identical parts as seen by the voter as $ObservedPart_b$.
8. \mathcal{A} opens commitment c_b and sends signed values of b and K_b to \mathcal{VM} which verifies both. \mathcal{VM} now knows the value of b .
9. \mathcal{VM} sends b to \mathcal{V} . The vote v is cast for the ballot and \mathcal{VM} publishes the signed pair $(s, code_{s,v})$ in the list of cast ballots on the public bulletin board. \mathcal{A} publishes the signed pair $(s, code)$ in the list of verified ballots on the public bulletin board. Discrepancies in the list are resolved by the examination of the public transcript between the two ITMs.

If a participant observes dishonest behavior by another participant, it aborts. When \mathcal{V} aborts, she sets $CPart := \text{“False”}$ (that is, she makes a complaint to a polling official). When \mathcal{A} aborts, it sets $CProof := \text{“False”}$.

The outputs are as follow:

1. The receipt:

$$\begin{aligned} \text{Receipt}(v, b) = & \\ & (\text{Receipt}_s, \text{Receipt}_E, \text{Receipt}_P) \\ & := (s, \text{code}, \text{Tape}_{\mathcal{A},\mathcal{VM}}) \end{aligned}$$

2. The check of the assistant: $\text{CProof} := \text{CheckProof}(\text{Receipt}(v, b))$ is “True” if \mathcal{A} publishes the signed pair $(\text{Receipt}_s, \text{Receipt}_E)$ in the list of verified votes, and is “False” else.
3. The ballot-part, sent by \mathcal{A} to \mathcal{V} : ObservedPart_b
4. The check of the voter: if $\text{CPart} \neq \text{“True”}$ then $\text{CPart} := \text{“False”}$.

7 Properties of the eCutAndChoose Protocol

In this section we provide rigorous statements for properties of the eCutAndChoose protocol, proof sketches may be found in the appendix. We model these properties on the more general ones for an AHIP in [11]. We assume that all cryptographic primitives used are secure and all adversaries are PPT.

Theorem 1 (COMPLETENESS). *If all three parties are honest, $\text{Receipt}_E = \mathcal{E}_s(v)$, $\text{CProof} = \text{“True”}$ and $\text{CPart} := \text{“True”}$.*

Lemma 1. *If $\text{Receipt}_E \neq \mathcal{E}_s(v)$ and \mathcal{A} and \mathcal{V} are honest,*

$$\Pr[\text{CProof} = \text{False OR CPart} = \text{False}] \geq \frac{1}{2} - \alpha$$

for some small value α

Lemma 2. *There is no means by which \mathcal{A} can change a vote without colluding with \mathcal{VM} .*

Theorem 2 (SOUNDNESS). *If at least one of \mathcal{A} and \mathcal{VM} is honest and $\text{Receipt}_E \neq \mathcal{E}_s(v)$,*

$$\Pr[\text{CProof} = \text{False OR CPart} = \text{False}] \geq \frac{1}{2} - \alpha$$

Theorem 3 (PRIVACY). *A public transcript for vote v is computationally indistinguishable from one for vote v' for all pairs (v, v') .*

We denote the protocol \mathcal{P} . Consider a dishonest \mathcal{A} , denoted Adv desiring to coerce voter \mathcal{V} to vote v . Suppose Adv designs a new interaction with \mathcal{V} during the protocol, instructs \mathcal{V} to vote v before or during the protocol, and queries \mathcal{V} after the protocol is over. The purpose of this would be to coerce \mathcal{V} into voting in a certain manner. Let us denote the new protocol \mathcal{P}' , and the view of original participant \mathcal{X} in \mathcal{P}' as $\text{view}_{\mathcal{P}', \mathcal{X}}$. Note that, from the perspective of \mathcal{VM} , the protocol is identical to \mathcal{P} . We denote by $\mathcal{V}_{\text{coerced}, \mathcal{P}'}$ the voter in \mathcal{P}' obeying the instructions of \mathcal{A} . We say a protocol is coercion-resistant if there exists a strategy \mathcal{V}^* for the voter which is indistinguishable from $\mathcal{V}_{\text{coerced}, \mathcal{P}'}$ for Adv in protocol \mathcal{P}' .

Theorem 4 (COERCION-RESISTANT). *If \mathcal{VM} is an honest participant in protocol \mathcal{P} , \exists PPT \mathcal{V}^* such that $\text{view}_{\mathcal{P}', \text{Adv}}$ when interacting with $\mathcal{V}^*(v')$ in protocol \mathcal{P}' is computationally indistinguishable from $\text{view}_{\mathcal{P}', \text{Adv}}$ when interacting with $\mathcal{V}_{\text{coerced}, \mathcal{P}'}(v)$.*

8 Prototypes

We have implemented proofs of concept which are fully functional for both `eChallengeResponse` and `eCutAndChoose`, and for both the voting machine and the personal assistant. Our implementation uses parts of the open-source Scantegrity II code. We use the Scantegrity II back-end implementation (which is actually a modified version of the Punchscan back-end [6]) to perform a verifiable-tally.

The prototype implements two modules: `DRE` (the voting machine) and `smartphone` (the computational assistant), which handle their own network communication.

The implementation of the `DRE` module is in Java 1.5, and has an interactive voice control system, as well as a synchronized visual interface. The two interfaces can be checked for consistency using observational testing. Full functionality for the `DRE` module has been tested to date on Microsoft Windows XP, Microsoft Windows 7 and Mac OS X Snow Leopard; there are some known problems on Linux systems that currently prevent full functionality. In our implementation, each instance of the `DRE` module posts confirmation codes directly online, to a secure public bulletin board (which we assume exists, but did not implement).

The `smartphone` module is implemented on an Android 1.6 platform, using Java 1.5 and the Bouncy Castle cryptographic library. We have successfully tested it on the HTC Hero and the Nexus One. Our security model assumes that the `smartphone` module take no input from the voter. For research and testing reasons, however, our current implementation allows the voter to choose whether to cast or audit (`eChallengeResponse`) or whether the left or right ballot half is opened (`eCutAndChoose`). As we describe in section 5, this has no impact on security results as long as the voter does not provide any other information to the `smartphone` module during the protocol. The application design assumes a touchscreen front-end and does not use any other type of input device (such as a hardware keyboard).

An essential part of the project is communication between `DRE` and `smartphone`. We connected the two using USB and we used the Android Debug Bridge, a tool contained in the Android SDK. Using `adb` we were able to initiate port forwarding on the host, meaning that all communication aimed at a certain port on the localhost is automatically forwarded to the `smartphone` through the USB connection. We then implemented sockets on top of this to achieve network communication between the two devices. Newer versions of Android, starting with 2.0, have added better support for networking over bluetooth so this could be a new approach in a future version.

In addition to standard libraries, we also used open source code from the Helios and Scantegrity codebases. To commit to a single bit, we use Pedersen bit-commitments, and to commit to larger messages we use Scantegrity commitments. We use the 1-2 oblivious transfer protocol of Even, Goldreich and Lempel [10].

8.1 Network Access

One of the major real-world threats for a voting system like ours is to be online. Voting machines can be hacked, and the leakage of cryptographic keys can lead to unauthorized ballot casting. On the other hand, if we choose to be offline we face two important issues.

First, the assistant cannot check if a given ballot is fresh. Second, the assistant is required to verify the opening of commitments against those published before the election.

A solution to the first problem is to pre-distribute ballots among voting machines. A voting machine that reuses a ballot is caught because it signs the serial number when it offers the ballot to the assistant. A solution to the second problem is for the assistant to maintain a root of a Merkle tree (hash tree) of the commitments so it can verify commitments offline. The voting machine is required not only to open audited commitments but also to show the hash path. This is preferable to storing all commitments locally on the assistant. It allows us to perform ballot casting offline at the cost of increasing the communication complexity between the voting machine and assistant by $O(\log N)$ where N is the number of ballots.

8.2 Security of the Implemented Protocols

We describe common security concerns in implementations. First, there is the possibility of an attacker tapping the communication between the devices. This is not a concern for us because the transcripts between the two devices are assumed public. Second, there is the possibility of an attacker changing the communication. All messages between the two devices are hence digitally signed. Third, the smartphone is not an output-only device. However, in our implementation, the smartphone will have to be in a special docking station while the protocol is on. In the station, it is protected by a transparent casing or sheet, so that the voter may obtain information from it, but not provide it any input. The voter can only provide input when it is not in the station, and if it is removed from the station the voting system aborts the protocol.

8.3 Ongoing Work

Scantegrity's back-end does not explicitly commit to the correspondence between candidates and confirmation codes; this correspondence may be deduced from other commitments made by the back-end. We are in the process of writing code to generate these commitments ourselves for use with `eChallengeResponse`. Since the system uses an electronic voting machine instead of a paper ballot, some sort of voter authorization mechanism should be used to ensure that each voter casts only one ballot, but is allowed to audit multiple ballots in `eChallengeResponse`. Such functionality has been discussed, but is not yet implemented—the voter would use a one-time password when she wished to cast a vote. An open-source release of the current version of our code will soon be complete.

9 Conclusions

We have described a paperless vote-casting protocol where the voter uses a computational assistant that participates in an interactive protocol with the voting machine. We propose the *Actively-Assisted Human Interactive Proof* model to study these types of protocols. We have rigorously described the AAHIP and demonstrated its security properties, assuming that the voting system and assistant do not collude and that communication is synchronous. The assistant may be malicious, however, and provide false values or

refuse to provide them. Its memory and logs may be examined by anyone after the vote is cast. We also show that a simple challenge-response-style AAHIP has weaker coercion-resistance. We have described our prototypes of both protocols, using a smartphone for the assistant.

Some interesting questions remain open. Is the cut-and-choose protocol secure against stronger adversarial models—for example, if communication is asynchronous? What is the most security one can obtain in the AAHIP model? Does the blind-signature-based protocol solve the problem of dispute resolution in paperless protocols?

References

1. Adida, B. *Advances in Cryptographic Voting Systems*. PhD thesis, MIT (2006)
2. Adida, B.: Helios: Web-based Open-Audit Voting. In: *Usenix Security Symposium* (2008)
3. Benaloh, J.: Simple verifiable elections. In: *USENIX/Accurate Electronic Voting Technology Workshop* (2006)
4. Benaloh, J.: Ballot casting assurance via voter-initiated poll station auditing. In: *USENIX/Accurate Electronic Voting Technology Workshop* (2007)
5. Benaloh, J., Tuinstra, D.: Receipt-free secret-ballot elections. In: *ACM Symposium on Theory of Computing* (1994)
6. Carback, R., Chaum, D., Clark, J., Essex, A., Mayberry, T., Popoveniuc, S., Rivest, R.L., Shen, E., Sherman, A.T., Vora, P.L.: Scantegrity II Municipal Election at Takoma Park: The First E2E Binding Governmental Election with Ballot Privacy. In: *Usenix Security Symposium* (2010)
7. Chaum, D.: Secret-ballot receipts: True voter-verifiable elections. *IEEE Security and Privacy*, 38–47 (January/February 2004)
8. Chaum, D., Carback, R., Clark, J., Essex, A., Popoveniuc, S., Rivest, R.L., Ryan, P.Y.A., Shen, E., Sherman, A.T., Vora, P.L.: Scantegrity: End-to-end verifiability for optical scan elections. *IEEE Transactions on Information Forensics and Security: Special Issue on Electronic Voting* 4(4), 611–627 (2009)
9. Chaum, D., Popoveniuc, S., Vora, P.L.: eTegrity and ePunchscan. In: *NIST End-to-End Voting Systems Workshop* (October 2009), http://csrc.nist.gov/groups/ST/e2evoting/documents/papers/Popoveniuc_PaperlessVoting.pdf
10. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. *Communications of the ACM* 28(6), 637–647 (1985)
11. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: *Workshop on Privacy in the Electronic Society, WPES* (2005)
12. Katz, J., Lindell, Y.: *Introduction to Modern Cryptography*. Chapman & Hall/CRC (2008)
13. Kelsey, J., Regenscheid, A., Moran, T., Chaum, D.: Attacking Paper-Based E2E Voting Systems. In: Chaum, D., Jakobsson, M., Rivest, R.L., Ryan, P.Y.A., Benaloh, J., Kutyłowski, M., Adida, B. (eds.) *Towards Trustworthy Elections*. LNCS, vol. 6000, pp. 370–387. Springer, Heidelberg (2010)
14. Neff, C. A.: Practical high certainty intent verification for encrypted votes (2004)
15. Popoveniuc, S., Hosp, B.: An Introduction to PunchScan. In: Chaum, D., Jakobsson, M., Rivest, R.L., Ryan, P.Y.A., Benaloh, J., Kutyłowski, M., Adida, B. (eds.) *Towards Trustworthy Elections*. LNCS, vol. 6000, pp. 242–259. Springer, Heidelberg (2010)
16. Ryan, P.Y.A.: A variant of the Chaum voter-verifiable scheme. Tech. Rep. CS-TR: 864, School of Computing Science, Newcastle University (2004)
17. Sandler, D.R., Derr, K., Wallach, D.S.: VoteBox: a tamper-evident, verifiable electronic voting system. In: *USENIX Security Symposium* (2008)

Feasibility Analysis of Prêt à Voter for German Federal Elections

Denise Demirel¹, Maria Henning², Peter Y.A. Ryan³, Steve Schneider⁴,
and Melanie Volkamer¹

¹ Technische Universität Darmstadt / Center for Advanced Security Research
Darmstadt, Germany

² Project Group Constitutionally Compatible Technology Design (provet),
Universität Kassel, Germany

³ University of Luxembourg/ Interdisciplinary Centre for Security and Trust,
Luxembourg

⁴ University of Surrey, United Kingdom

Abstract. Prêt à Voter is one of the most well-known and most extensively analysed electronic voting systems for polling stations. However, an analysis from a legal point of view has not yet been conducted. The purpose of this paper is to analyse the readiness of Prêt à Voter for legally binding federal elections in Germany. This case is of particular interest as Germany has with the Constitutional Court Decision from 2009 probably the most restrictive requirements on electronic voting in particular regarding the public nature of elections and verifiability respectively. While many aspects are analysed, some remain open for further legal and technical discussions. Thus, a final decision is not yet possible. Aspects analysed are the ballot paper layout, different processes from ballot printing through to the publishing of results, as well as verifiability, and the overall election management.

Keywords: Verifiable Elections, Legal Requirements, German Federal Elections, Prêt à Voter, Election System Design.

1 Introduction

Since the 1960s several attempts have been made in Germany to replace manual casting and counting of votes by mechanical and later electronic voting systems. These efforts have always been based on the ambition to obtain a correct election result within a very short period of time. Electronic voting machines were first deployed in Germany on the occasion of the European elections in 1999. These machines produced by the company of Nedap were used on all election levels. After their usage for the elections of the German Bundestag (Federal Diet) in 2005 two people filed a complaint against this election. These complaints went through several instances and ended up at the Federal Constitutional Court. In 2009, the Federal Constitutional Court declared the used electronic voting machines and the Federal Voting Machine Ordinance which defines the requirements a voting machine has to ensure, to be unconstitutional because both did

not meet the requirements emerging from Article 38 in conjunction with Article 20.1 and 20.2 of the German Constitution. From these articles the court deduced that it must be possible to check the essential steps in the election process including the accurate counting of votes [5, page 71]. Now, election authorities are looking for an electronic voting machine that meets these requirements and can therefore be classified as constitutionally compatible.

Prêt à Voter is one of the best known electronic voting systems which implements verifiability, has a prototype actually implemented, and has been successfully used in (test) elections in the U.K. Over time different variations of Prêt à Voter have been published and their security has been analysed. However, an analysis from a legal point of view has not yet been conducted.

This paper analyses which version of Prêt à Voter fits best to the regulations of the Federal Electoral Act, the Federal Electoral Code, and the German Constitution. It also discusses necessary modifications to the system and the processes. In general, we tried to keep the process for voters as similar as possible to what they know and as simple as possible. Such an analysis was possible due to the cooperation and interdisciplinary work between computer scientists and legal researchers. We categorise our discussion into the following groups: aspects that are relevant for the ballot paper design, the different processes, verifiability, and election management. Voting registration and authentication in the polling station are not taken into account for this paper as these processes do not need to be modified.

The paper is structured in the following way: we first provide an introduction to the German federal elections in Section 2 and then to Prêt à Voter in Section 3. Afterwards, we discuss legal and technical aspects of the ballot paper design in Sections 4 and 5, different processes starting from ballot printing until the publishing of receipts in Section 6, the type of verifiability in Section 7 and the overall election management in Section 8. The paper concludes with a brief summary and remarks for future work (Section 9).

2 German Federal Election

According to Article 38.1 German Constitution/Grundgesetz (GG) the Bundestag is elected in universal, direct, free, equal and secret suffrage. Subject to particular regulations, the elections take place every four years. Voters can cast two votes on one ballot paper (see Fig. 1). With the first vote they select a named candidate from their home county (electoral district). With the second vote they select the list of a party on state level. Half of the delegates move into parliament because of the first vote, half because of the second vote. After §1.1 of the Federal Electoral Act (FEA), the Bundestag generally consists of 598 delegates.

According to [6], roughly 62.2 million Germans were eligible to vote in the elections for the Bundestag in 2009. 70.8 percent cast their votes while 1.7 percent of the first votes and 1.4 percent of the second votes were invalid. Based on the handover of empty or crossed ballot papers, it is assumed that nearly 70 percent of them have been spoiled on purpose. For the elections of the Bundestag

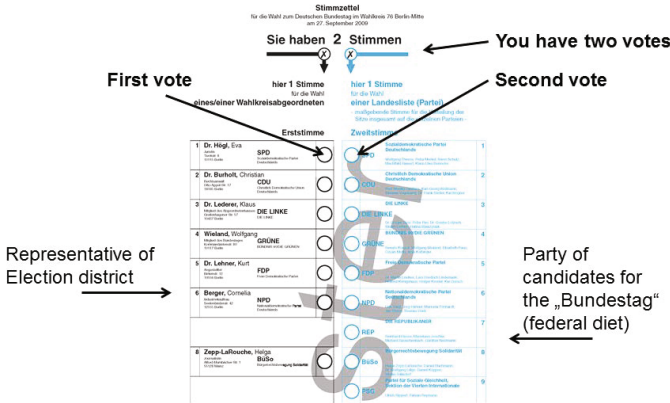


Fig. 1. Ballot paper for the German Federal Election in 2009

the territory of the 16 federal states is subdivided into 299 electoral districts. Due to the fact that every electoral district nominates local candidates for the first vote, the ballot papers differ from one district to another. The electoral districts are subdivided into constituencies. After §12.1 of the Federal Electoral Code (FEC) municipalities with no more than 2,500 inhabitants (generally 1,700 voters) usually form one constituency. Altogether there were 75,081 constituencies in Germany in 2009.

According to §31 FEA and §54 FEC everyone (not only eligible voters but everyone who is interested) is allowed to be present at the polling station during the whole election procedure and during the vote counting as long as they do not disturb any processes. Thus, people can observe the correct operation of the election and the correct counting of votes. After §47.1 FEC the polling stations are open from 8am until 6pm on a Sunday. The ascertainment of the results starts right after closing the polling stations. After deciding about the validity of every vote, the returning committee¹ ascertains the votes cast in the constituency according to §§67–71 FEC. Referring to this, the head of the returning committee informs the local authority and the district election officer who passes the respective district results to the state returning officer. The state returning officer refers the results directly and continuously to the federal returning officer who publishes the provisional curatorial election result. In the elections for the Bundestag in 2009, this result was published at 3.25am Monday morning.

3 Prêt à Voter

Prêt à Voter is an end-to-end verifiable voting system. It provides secrecy of the ballot and integrity of the election, and is designed also to allow verification of

¹ The returning committee is the electoral body who takes care of the ordinary run of the election, watches the observance of the electoral principles and counts the votes after closing the election.

the processing of the votes, from casting through to tallying. It achieves this by publishing auditable information for each stage the votes pass through, so that verification rests on the information that is published rather than the processes which generated that information. In fact there are several versions of Prêt à Voter, which vary in terms of the detail of how this is achieved, and an overview of the differences and of the general Prêt à Voter approach is given in [11]. In this paper we focus on elections where a vote is cast against a single candidate. Variants of Prêt à Voter are also able to handle preferential voting, but we will not consider them here.

The key idea is that Prêt à Voter sets up the voting process so that voters cast their vote in a familiar way, but the system accepts their vote in encrypted form. This allows publication of the list of encrypted votes cast, and also allows the voters to retain a record of their cast vote to confirm that it appears on the published list. The system then anonymises the votes, decrypts them, and finally tallies the results. The anonymisation phase means that no decrypted vote can be linked with any encrypted vote cast by a specific voter, preserving ballot secrecy. There are verification mechanisms for each of these phases—anonimisation, decryption, tallying—which mean that the integrity of the election can be verified. A diagrammatic overview of the process is given in Figure 2.

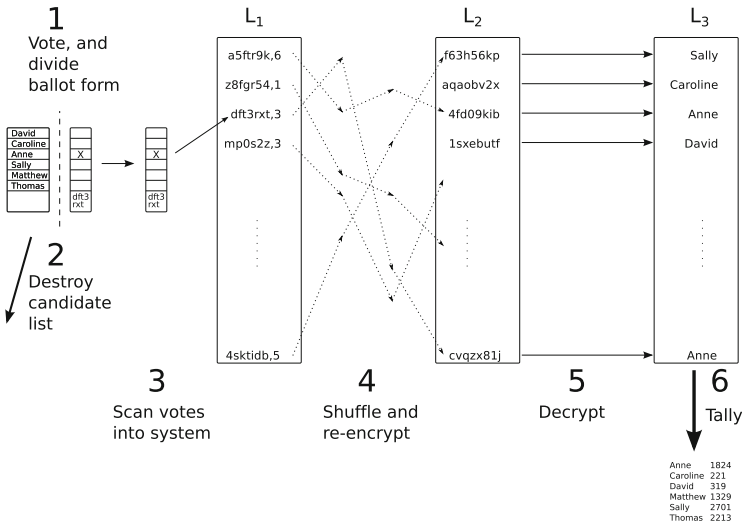


Fig. 2. Overview of the phases of Prêt à Voter vote processing

3.1 Vote Casting

The central idea of Prêt à Voter is the use of a particular design of ballot form to capture the vote. In the proposals to date, each ballot form contains the list of candidate names on the left hand side, in a random order which varies between ballot forms. The right hand side has a space against each candidate for the

voter to mark their vote, and also (at the bottom) the order of the candidates in encrypted and hashed form². The ballot form is perforated to allow the two sides to be separated. The voter marks their vote against their chosen candidate, and then separates the two halves of the ballot paper and destroys the list of candidates on the left hand side. The right hand side contains a vote marked in some position, and the hash value of the ciphertext containing the order of candidates³. The right hand side is then scanned into the election system, the scanner's interpretation (in terms of the position of the mark) is displayed to the voter, the voter confirms this is correct (or else corrects it either on the display or by filling out a new paper ballot), and it is then accepted by the system for inclusion in the published list L_1 of cast votes, which is published on an online Web Bulletin Board (BB) together with the encryption of the corresponding candidate order. The system will only accept a vote in which exactly one selection is made. The system provides the voter with a signed record of the encrypted vote that has been accepted, which contains the cryptographic information on the original right hand side, and the marked position. This can be compared with the one the voter scanned. While the original one is put into a ballot box the printed and signed one is kept by the voter as receipt. The voter can later check it against the published list of votes cast by looking it up on the BB. The record does not reveal which candidate received the vote, since the random order of candidates means it could have been any of them.

3.2 Vote Processing

The list of accepted scanned votes L_1 is a list of encrypted votes. They are now processed through a series of stages to yield the final tally. The first stage is *anonymisation*: this is achieved by passing L_1 through a re-encryption mixnet [10], which shuffles the encrypted votes through a series of *mixnet nodes* which perform secret permutations, while re-encrypting the votes. The result is a new list of different ciphertexts which encrypt the same votes as the original list but in a different order. This resulting list L_2 is also published. No vote in L_2 can be linked to any vote in the original list, and hence cannot be linked to any vote record held by a voter. The shuffling and re-encryption can be verified, either by randomised partial checking [8], or through proofs of re-encryption [13,9], depending on the approach taken. Several mix servers are used, each to perform a round of re-encryption and shuffling: secrecy of the shuffle is obtained provided they do not all collude.

The second stage is *decryption*. The resulting list L_2 of encrypted votes can now be decrypted. The decryption key is shared across the election servers, so that a minimum number (threshold set) of them are required to cooperate to decrypt the ballots. The resulting list L_3 is also published. The decryption can be audited against L_2 , and verified as valid by the public.

² The hash is used to keep it short. Instead of the hash value also a serial number could be used while this number is bounded to the ciphertext of the candidate order e.g. via commitments of the Web Bulletin Board.

³ In the following we will use the term hash value when referring to this ciphertext.

The final stage is *tallying*. The decrypted votes are tallied in order to obtain the result of the election. Since the tallying algorithm is public, and the decrypted votes are published, the tallying process can be checked and verified by any party.

4 General Legal Analysis of Random Candidate Order

In this section we analyse whether shuffling candidates complies with the legal regulations for federal elections in Germany. After §30.3 FEA the order of the party lists on state level depends on the number of the second votes each party achieved in the particular federal state within the last election. The parties that are not currently represented in parliament follow in alphabetical order. The order of the county proposals presented on the left side of the ballot paper depends on the order of the respective party list. Other county proposals follow in alphabetical order.

Correspondingly the random order of Prêt à Voter is not compatible with the current regulation of the FEA. However, the question if it might be permissible in general, depends on constitutional requirements because the respective regulation can be changed if this is constitutionally compatible. The Federal Constitutional Court abnegated the obligation to provide equal ballot papers in its decision from the 6th October 1970 [4, p. 164]. It pointed out that the regulation relevant to the case ensures the ordinary flow of the election, but is not necessary from a constitutional point of view.

The principle of the equal suffrage according to Article 38.1 GG forms the basis for the organisation of the election and the functionality of the voting system [14, §1, Rn. 42]. After the jurisdiction of the Federal Constitutional Court it contains the equality of counter value⁴ for the first and second vote, the equality of result value⁵ for the second vote and the equality of opportunity for all candidates [1, p. 246, 247].

The third aspect would be strengthened by the random order. In reference to this every candidate needs to have the same chance to win the election. Although a strict order is not said to violate the principle of the equal suffrage because voters prefer to make their decisions based on the manifesto of the political party [2, p. 18], it brings a psychological benefit to the party that achieved the highest number of second votes within the last election and therefore comes up on the top of the race [14, §30, Rn. 8]. This effect would be lost by using the random order of Prêt à Voter. Therefore, non-established parties would win the benefit to be on top of the list periodically. Consequently, the key idea of Prêt à Voter brings a big advantage over the known paper ballot system. New parties could obtain the benefit of getting registered first by the voters just like the catch-all parties. Through this the constitutional requirement of the equal suffrage would be invigorated. Furthermore, a strict order of the candidates is not compulsory according to the election principles of Article 38.1 GG.

⁴ Every person, eligible to vote, has the same number of votes and is able to vote in the same way.

⁵ All votes have the same influence on the election result.

5 Analysis of Design Proposals — Ballot Paper

In this section, we discuss different aspects of the ballot layout including which of the possible types of randomization for candidates and parties is most appropriate, how to design the ballot paper to easily remove the part that is scanned, how to integrate the hash value of the encrypted order of candidates, and how to enable invalid votes.

5.1 Type of Random Order

According to [11], two different types of randomisation can be applied, namely completely arbitrary order or cyclic shifting of candidate order. Chaum et al. point out in [7] that the distance between marks on the receipt can leak some information if entries are only shifted and the voter is allowed to mark more than one entry. This is not the case for arbitrary order of entries as the successor and predecessor of every entry differs from ballot paper to ballot paper. The current ballot paper for the German federal election contains two races on one paper in two columns (compare to Fig. 1) which leads to a similar problem then mentioned in [7]. For instances, two entries in a row leak the information that the voter cast a vote for a direct candidate and the associated party which would violate the principle of the secret suffrage according to Article 38.1 GG. This principle includes that it needs to remain secret whether voters split their votes or cast them based on a single preferred party [14, §1, Rn. 95].

Cyclic shifting has, compared to an arbitrary order, the advantage that it is easier for the voter to find their candidate and party since the sequence of the candidates stays the same on all ballot papers. Therefore, the most appropriate way to randomize the order seems to be the use of cyclic shifting while each race is shifted with its own value. Note, in case where one race has more entries (candidates or parties), the empty columns are not included in the shifting algorithm (compare to Fig. 3). This approach leads to two different shift values which can either be encrypted together to one ciphertext and then hashed or the hash of two separate ciphertexts is computed.

5.2 General Ballot Layout

Using Prêt à Voter it is necessary that the voters can easily detach their markings and destroy the parts with the candidates on. On the current ballot paper for federal elections the candidate list and party list respectively shows up on the left and right hand side and voters mark their decision for both races in the centre of the ballot paper. As approaches like using two different ballot papers, putting the second race below the first race, and swapping the two races are either not practical or do not comply with §30.2 FEA and §45.1 FEC including appendix 26 of FEC respectively, we recommend to keep the ballot paper format from the traditional system but prepare it in a way that it is easy to detach the candidate lists on the left and right hand side. This remaining central part of the ballot paper, containing the marked positions and the hash value is scanned.

5.3 Obscuring Hash Values on Ballots

Although voters are not committed to keep their voting decision secret, they are not allowed to get a receipt of it which can be used as an evidence of the respective vote. Due to the possibility that voters might take a picture of themselves including the marked but not detached ballot paper and the respective hash value which will be published on the BB later, the secrecy of the vote and indirectly also the free suffrage of the election (compare to [14] §1, Rn. 94] could be violated. The picture together with the corresponding entry on the BB serves as a proof to sell votes. By entering the code on the BB, a potential extortionist could control if voters really cast their vote, which position they marked and correspondingly (by using the picture) for which candidate. Note, also in the traditional system one could take a similar picture but this picture does not serve as proof as the voter might have asked for a new ballot paper afterwards. Correspondingly, the hash value may not be plain as long as it is possible to link it to the particular ballot form and the order of entries on it. This is an issue during the vote casting process.

To solve this problem, the hash value could be hidden by various measures e.g. a scratch strip or invisible ink. Organisational procedures need to ensure that voters first detach the centre part of the ballot, destroy the remaining parts in the polling booth and then reveal the covered hash value outside of the polling booth before scanning it. While this is required from the legal perspective more research on this approach is required including whether voters would feel comfortable with the strict process needed in this case and technical challenges.

5.4 Enabling Invalid Votes

Due to privacy issues Prêt à Voter neither accepts under- and over-votes nor empty votes. In the traditional system any of these possibilities can be used to cast an invalid vote. Also paintings or writings on the ballot paper would result in an invalid vote. If a corresponding vote is scanned by Prêt à Voter the system will inform the voter and reject this vote.

§39 FEA defines the term of invalid votes and ascertains the way to deal with them. According to [14] §1, Rn. 23], the principle of the free suffrage contains the right to cast invalid votes. Otherwise people eligible to vote could be forced to cast a vote for a candidate. In reference to this, voters are allowed to vote the candidate they like, vote invalid while participating in the election, and to abstain from voting as well. In Prêt à Voter this aspect can be addressed by adding one field to the candidate/party list of every race with the option to cast an invalid vote (as proposed in [15]).⁶

From a legal perspective, it needs to be discussed whether voters can be forced to vote invalid in a certain way by marking the respective field. This might violate the principle of the free suffrage. But electronic voting is discussed in order to

⁶ Note, in order to spoil the whole ballot paper it is required to mark both invalid vote entries.

reduce the number of invalid votes cast by accident. The voting system of Prêt à Voter is able to support this intention.

All the different aspects of the ballot designed have been combined in an example ballot shown in Fig. 3.

German Federal Election 2009			
Invalid vote	<input type="radio"/>	<input type="radio"/>	Party E
Candidate A	<input type="radio"/>	<input type="radio"/>	Party F
Candidate B	<input type="radio"/>	<input type="radio"/>	Invalid vote
Candidate C	<input checked="" type="radio"/>	<input type="radio"/>	Party A
Candidate D	<input type="radio"/>	<input checked="" type="radio"/>	Party B
	<input type="radio"/>	<input type="radio"/>	Party D

Fig. 3. Proposed ballot paper

6 Analysis of Design Proposals — Processes

In this section, we discuss the relevant processes including ballot printing, vote casting, scanning, and the publishing of the receipts on BB.

6.1 Ballot Printing Process

There are two possibilities to consider for the timing of printing the ballot forms. It can be carried out either on demand in the polling station, or in advance of Election Day. The advantage of printing on demand is that there are no chain of custody issues with respect to the physical ballot forms, since they do not exist in physical form before they are printed for immediate use. However, printing on demand has a number of disadvantages: printers are needed in each polling station and additional effort at local level is required. The inclusion of measures such as scratch strips or invisible ink to mask the hash value will require special equipment. Hence, printing on demand is not appropriate. We therefore recommend that printing of the ballot forms is carried out in advance of the election.

We further recommend that the Election Manager takes responsibility for the printing at the same physical location as the generation of the ballot forms, where they are printed directly as they are generated, so that no electronic file of the ballot forms needs to be created. The ballot form generation system (servers and printers) should also be isolated from any network. A high level of trust is required in the printing provider, since it knows the association of ballot orders and the hash values. This can be mitigated by involving several parties in the printing process and using mechanisms to distribute the information between them, as discussed in [12], though this can become cumbersome.

Once generated, the chain of custody of the ballot forms through to their use in the election must be securely managed, to ensure that the information they contain remains secret, and that ballot forms cannot be added or removed.

6.2 Vote Casting Process

The detached centre part of the ballot form could either be scanned inside or outside of the polling booth. In case of the first variant, voters would have to scan the respective part on their own and without any external help. Irrespective of the question if they might be able and willing to do so, the returning committee could not confirm that the hash value is not revealed until the scanning process starts. This is no option due to the possibility of violating the principle of the secret suffrage (see Section 5.3).

In the second case, voters would have to scan the detached centre part of the ballot paper after having revealed the hash value outside the polling booth in front of the returning committee. Because voters might appreciate the help given by the returning committee, this approach would probably be the more pleasant one from a practical point of view. According to [14, §1, Rn. 95] nobody shall know if voters cast an over, under or empty vote if they do not disclose their voting decision by themselves. It should be noted that the system reveals some information by rejecting the scanned paper.

However, this situation could be improved for voters by providing a separate scanner in the polling booth. Voters could run a test scan before casting their votes outside in order to check whether the system accepts their ballot and whether they agree with the interpretation of the scan. While changing the rules for casting a vote change (in particular for casting an invalid one), we expect people to feel more comfortable pre-scanning their ballot in the voting booth.

Voters could still try to scan an invalid vote in public, disclaim the possibility of checking their filled ballot paper in advance, add marks after they checked the ballot paper or just ignore the feedback. But this situation might be comparable to the one regulated in §56.6 and 8 FEC. After this the returning committee has to rebuff every voter who tries to put an unfolded ballot paper into the ballot box. In case the voter still wants to cast a vote, the returning committee hands out a new ballot form after destroying the old one. Herewith the legislator wanted to assure voters do not waive their right of secret voting. However, further research from a legal point of view is required to back up this comparison.

6.3 Vote Scanning Process

Once the strip with the marks is scanned, there exist several possibilities to interpret the marks as votes. For instance one can detect crossing lines or the degree of blackening in the bubbles. The interpretation algorithm must comply with the legal guidelines.

After §34.2 of the FEA voters cast their votes by marking the particular section of the candidate on the ballot form with a cross or any other sign which indicates the voting decision. Consequently voters are free to choose any mark

they like as long as it is not unconstitutional. The secret suffrage for example is violated by using a very individual sign like signatures, which could be attributed to a particular voter. A sign outside of the field is not forbidden if an allocation to a certain candidate is possible [14, §34, Rn. 4]. But according to §39 FEA the votes cast are invalid in certain cases, e.g. when the voter hands in a ballot paper with an addition or a caveat on it. Although both terms are not defined in the FEA, the returning committee needs to decide about the validity of every vote. Therefore decisions can be made differently in different constituencies. After [14, §39, Rn. 12] for example an addition is every verbal annexation on the ballot paper which outruns the allowed annexation according to §34.2 FEA. Admittedly non-verbal terms need to be covered as well, e. g. a skull and crossbones on the ballot paper.

The scanner used by Prêt à Voter needs to be programmed in order to convert the given information while taking these uncertainties into account. Note, as voters still cast their vote on ballot papers it is possible that they could add marks or pictures which make the vote invalid but would be accepted by the system. The system's interpretation of the scan will naturally depend on the thresholds used, for example the degree of blackening accepted as a mark, in the image processing algorithms applied to the scan, and this may differ from the voter's expectation. Therefore the voting system of Prêt à Voter provides a confirmation stage for the voter. This option would show voters how their vote is interpreted, and provide the option to fill in another ballot paper if required. The confirmation stage would solve another problem as well: if voters put a comment into the section for a candidate — even the comment shows the antipathies of the voter — the system would count this as a valid vote for the respective candidate. This can only be acceptable when the voter confirms the interpretation of the system. On the condition of this confirmation stage, it might be legally compatible that the scanning process ignores additional marks or pictures and accepts ballot papers where exactly one mark per race is detected. Further this solution provides a consistent processing of votes.

6.4 Process of Publishing Receipts

The scanner prints out a signed receipt for each voter to take home. In order to enable voters to verify that their encrypted votes are recorded correctly, the corresponding electronic version of these receipts are published on the BB. These receipts could be published right after casting the vote or after the official end of the election. If they are published during the Election Day, an internet connection in the polling station would be needed. Irrespective of the increasingly higher costs, the risk of manipulation over the internet would come up as well. Therefore, we recommend to take the equipment first to some central places in the election district, count and tally votes, get backups of everything and then publish the receipts containing the position of the marks and the hash value together with the corresponding ciphertext.

The publishing of receipts after the closing of the polling station is also compatible with the current regulations of the FEC. According to §§54, 67 FEC

voters can watch the counting of votes in traditional paper based elections only after the closing of the election. In addition, by publishing afterwards, voters would not lose the possibility to appeal against the election or certain election decisions. According to Article 41 GG complaints requesting the scrutiny of an election need to be discussed and adjudicated by the Bundestag. After §2.4 of the Law on the Scrutiny of Elections the complaints have to be submitted within two months after the elections. After the jurisdiction of the Federal Constitutional Court the unobstructed run of parliamentary elections requires that legal control during the election is reserved for complaints requesting the scrutiny of the election afterwards [3].

7 Analysis of Design Proposals — Verifiability

After the verdict of the Federal Constitutional Court from the 3rd March of 2009 the voter himself or herself must be able to verify whether his or her vote as cast is properly recorded as a basis for counting [5, page 72]. It is not sufficient if voters must rely on the functionality of the system without the possibility of personal inspection [5, page 72]. Electronic voting systems need to provide a possibility for the voter to check the essential steps in the election act and in the ascertainment of the results to the same amount as in traditional paper based elections. Here voters are able to watch the entire election procedure — from the opening of the polling station until the vote counting and tallying. Thus, even though only very few people take the opportunity to observe the whole process it is in general possible.

In this section we consider the verifiability aspects of the stages of processing the votes while we distinguish between individual and public verifiability and discuss their compliance with the demands from the court decision.

7.1 Individual Verifiability

Individual verifiability covers well-formedness (to ensure that votes are cast as intended) and the fact that the voter can verify that their hash value appears on the BB (to ensure that votes are stored as cast). *Well-formedness* of the ballot forms requires that the printed candidate list matches the list embedded in the hash value. Ballot forms can be checked by requesting a decryption of the ciphertext belonging to the hash value from the election decryption servers. There will be some random well-formedness checks by independent auditors ahead of and during the election. Due to the legal requirement this event will be announced and will be accessible by the public [7].

Also during the election, voters themselves may request to audit ballot forms that they are given. When given a ballot form, a voter can choose whether to select it for audit, or to use it to cast a vote. In the case of audit, the voter retains an audit copy of the entire ballot form with the visible hash value, and

⁷ In addition, to enabling them to observe even they cannot be physically present, a live stream could be broadcasted.

the system retains and logs that ballot form as selected for audit by entering the hash value in a separate interface. This is necessary to prevent the form also being used to vote, since secrecy will be lost. In the post election phase, the audit process will be completed, by decrypting and publishing the list of candidates, for the voter to verify.

If a vote is cast, the voter is provided with a *receipt* of the right hand side, and after the election when the receipts are published, the voter can use this to verify that the information contained on it has been included on the published list of cast votes, which will next become mixed and decrypted.

If voters detect a problem with either of these two lists, then the voter's receipt and the entire ballot form respectively are evidence that can be used as a basis for a challenge to the election. It is important that receipts and the entire ballot form for auditing cannot easily be forged, since that would enable voters to mount fake challenges.

7.2 Public Verifiability

The cast votes are submitted in encrypted form. The next stage of the process is to pass them through several rounds of a re-encryption mix, where each round re-encrypts and shuffles all of the votes. The output of that process is another list of encrypted votes, which can then be decrypted and tallied.

Re-encryption mixes allow several approaches to verification, as described earlier. The legal requirement that it must be possible to check the essential steps in the election process inclines us towards the randomised partial checking approach, rather than using proofs of re-encryption: the mechanism of random sampling and checking is more intuitive and comprehensible to the public, and in principle any observer can contribute to the random audit checks. Each stage of the mix, i.e. each intermediate list of encrypted votes, is made public. A check involves challenging a particular re-encryption link in the mix and obtaining evidence of the re-encryption (i.e. the randomisation introduced in that step) that can be independently checked. In a mix, half of the links will be randomly checked, but in a way that ensures that no receipt can be traced through the mix. More precisely, the audits, while essentially random, can be carefully constrained to ensure that there are numerous breaks in the chain from receipt to decrypted vote.

Several re-encryption mixes can be run, in parallel or at any later time on request, which on decryption of the output should all yield the same result. The confidence level can be made as high as required by adding parallel mixes and audits. For a winning margin of n votes, the probability that a different candidate was in fact the winner but no altered votes were found by the audit, will be at most $2^{-n/2}$, which decreases exponentially as n increases: for example, a winning margin of only 40 votes is 99.9999% certain to be correct if the randomised partial checking audit on a single re-encryption net is successful, i.e. does not find any incorrect re-encryptions. If a higher level of certainty is required, then further mixnets can be generated and audited.

The remaining steps of the process are all publicly verifiable: the decryption of the outputs from the mix along with proofs of correctness of the decryptions is published, and can be verified by anyone. The tallying of the decrypted votes is also published and can also be checked independently.

With this proposed verifiability of the election, the level of verifiability can be increased compared to the level of verifiability in traditional paper based election systems. However, what remains for future work is to deduce an acceptable certainty for both individual and public verifiability from both legal requirements and the accepted error rate in traditional paper based elections.

8 Analysis of Design Proposals — Election Management

The election can be run centrally, or it can be distributed. The advantage of running the election from a single central location is that the system needs to be set up only once, and one single election system is likely to reduce costs in terms of equipment and management in comparison to a number of smaller ones. However, the disadvantage is that a single system will be a bottleneck, both in terms of setting up the election in the first place, including the generation and printing of the ballot forms, and in terms of the length of time needed to process the large number of cast ballots.

Alternatively the election may be decentralised, and run across a number of locations. This can be achieved by setting up completely independent Prêt à Voter systems, each to run the election for a number of constituencies, and collating their results. This has the advantage of processing the cast votes in parallel, obtaining the result more quickly than a single central system would be able to. Another advantage of decentralisation is that challenges can be handled more efficiently at a more local level.

The election naturally separates into smaller races which can be run and processed separately: The ballot forms differ for every electoral district, so the preparation before the election is different for each district and can be distributed. Furthermore, the results of each electoral district must in any case be reported separately, meaning that the votes from each electoral district must be processed and tallied separately. The Prêt à Voter system is able to manage this by processing the votes in batches and reporting separately on the results, one district at a time, and this task distributes naturally across a number of such systems.

Our recommendation is therefore to decentralise the election management. The optimal level of decentralisation balances the overall resources required against the gains achieved in terms of efficient management of the election and speed of reporting the result, and in this context we would recommend decentralisation to the level of federal states.

9 Conclusion and Future Work

The paper analyses Prêt à Voter regarding its readiness for German federal elections and discusses different design proposals in order to decide which of the

many different variations of Prêt à Voter fits best. While already a lot of different aspects from the legal perspective are covered and ensured by the proposed version, some others need to be discussed in future work. The main one is the requirement of [5, page 39] that all essential steps in an election are subject to public examinability (unless other constitutional interests justify an exception). This needs to be possible without any special expert knowledge [5, page 39]. Against this background, it needs to be discussed whether and how a system like Prêt à Voter can fulfil this requirement. Therefore, it is first necessary to further analyse the judgement regarding the question whether voters need to be able to check the election manually or whether tool support is acceptable as well as understanding the general idea, or whether it is necessary to follow all mathematical steps (which would obviously not be possible for a system like Prêt à Voter). Besides this, future work includes discussions about data storage (which data for how long) and responsibilities (who prints and keeps the ballot papers, how many key holders, election servers, and mix nodes as well as who they are). Although the receipt gives no information about the particular voter, long term secrecy needs to be broached as well. Furthermore, an acceptable certainty for both individual and public verifiability needs to be deduced from legal requirements and the accepted error rate in traditional paper based elections. Besides that we have to view the fact that visually handicapped people cannot cast their votes by using a plastic template (with Braille on it to put the ballot paper in) anymore when introducing Prêt à Voter. Therefore, it needs to be discussed whether a particular amount of ballot papers with Braille needs to be available in each polling station.

In addition to these legal aspects, there are also usability and acceptance issues, for instance regarding whether detaching is feasible without destroying the centre part of the ballot paper and destroying the other parts, as well as accepting that only test ballots can be audited but not the one to cast and the strict processes demanded in the polling station.

Another challenge is extending a suggestion regarding the vote scanning process and enabling voters to check the signature on their receipt, to ensure their receipt is genuine evidence of their cast ballot. This requires additional equipment, perhaps provided by independent organisations. If this is not feasible then it may be appropriate to use conventional anti-counterfeiting measures, such as special paper for the receipts, special patterns, rubber stamping, and so on. This challenge is the subject of current research.

This paper shows that it is worth analysing systems with an interdisciplinary team to ensure that they are not only secure from a cryptographic point of view but also conform to elections laws for particular elections.

Acknowledgements. This paper has been developed within the project 'VerKonWa' — Verfassungskonforme Umsetzung von elektronischen Wahlen — which is funded by the Deutsche Forschungsgemeinschaft (DFG, German Science Foundation) and conducted in cooperation of provet (Project Group Constitutionally Compatible Technology Design at the University of Kassel) and CASED (Center for Advanced Security Research Darmstadt). Peter Ryan thanks the

FNR Luxembourg for funding the SeRTVS project. Steve Schneider is grateful for funding through the Trustworthy Voting Systems project under UK EPSRC grant EP/G025797/1.

References

1. Bundesverfassungsgericht: Judgment. BVerfGE 1, 208–261 (April 1952), <http://sorminiserv.unibe.ch:8080/tools/ainfo.exe?Command=ShowPrintVersion&Name=bv001208>
2. Bundesverfassungsgericht: Judgment. BVerfGE 13, 1–20 (May 1961), <http://sorminiserv.unibe.ch:8080/tools/ainfo.exe?Command=ShowPrintVersion&Name=bv013001>
3. Bundesverfassungsgericht: Judgment. BVerfGE 14, 154 (June 1962)
4. Bundesverfassungsgericht: Judgment. BVerfGE 29, 154–165 (October 1970)
5. Bundesverfassungsgericht: Judgment. BVerfGE 123, 39–88 (March 2009), http://www.bverfg.de/entscheidungen/rs20090303_2bvc000307en.html
6. Bundeswahlleiter, D.: Wahl zum 17. Deutschen Bundestag am 27. September 2009, Heft 5, Textliche Auswertung der Wahlergebnisse (November 2010), http://www.bundeswahlleiter.de/de/bundestagswahlen/BTW_BUND_09/veroeffentlichungen/Heft5_komplett.pdf
7. Chaum, D., Ryan, P.Y.A., Schneider, S.: A Practical Voter-Verifiable Election Scheme. In: De Capitani di Vimercati, S., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 118–139. Springer, Heidelberg (2005)
8. 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)
9. Neff, C.A.: A verifiable secret shuffle and its application to e-voting. In: ACM Conference on Computer and Communications Security, pp. 116–125 (2001)
10. Park, C., Itoh, K., Kurosawa, K.: Efficient Anonymous Channel and All/Nothing Election Scheme. In: Hellese, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 248–259. Springer, Heidelberg (1994)
11. Ryan, P.Y.A., Bismark, D., Heather, J., Schneider, S., Xia, Z.: Prêt à Voter: a voter-verifiable voting system. IEEE Transactions on Information Forensics and Security 4(4), 662–673 (2009)
12. Ryan, P.: Prêt à Voter with Paillier encryption. Journal of Mathematical and Computer Modelling 48(9-10), 1646–1662 (2008)
13. 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)
14. Schreiber, W.: Bundeswahlgesetz Kommentar. Carl Heymanns Verlag (March 2009)
15. Xia, Z., Schneider, S.A., Heather, J., Ryan, P.Y.A., Lundin, D., Peel, R., Howard, P.: Prêt à voter: All-in-one. In: Proceedings of IAVoSS Workshop on Trustworthy Elections (WOTE 2007), Ottawa, Canada, pp. 47–56 (2007)

Prêt à Voter with Write-Ins

Steve Schneider, Sriramkrishnan Srinivasan, Chris Culnane,
James Heather, and Zhe Xia

University of Surrey, Guildford, Surrey, GU2 7XH, UK

Abstract. This paper presents an extension of the Prêt à Voter verifiable voting system to handle write-ins. This is achieved by introducing an additional ‘Write-In’ option and allowing the voter optionally to enter a write-in candidate of their choice. The voter obtains a receipt which includes their write-in, but that receipt does not indicate whether the write-in candidate was selected or not. The system provides flexibility with respect to the tallying of write-in votes. We also introduce null ballots in order to achieve receipt-freeness with respect to write-ins.

Keywords: end-to-end verifiable voting, Prêt à Voter, write-in votes, receipt-freeness.

1 Introduction

End-to-end verifiable voting systems provide mechanisms for verifying elections. These generally involve checking cast votes against some evidence provided to the voter, such as a receipt of voting, or a code number. ‘Receipt-freeness’ is an important property of such systems. Receipt-freeness requires that the voter’s evidence, together with the evidence of integrity of the election provided by the authorities, should not give away how the voter voted. This protects against vote selling and voter coercion.

Elections allowing write-in candidates pose a particular challenge for end-to-end verifiable voting systems. A voter requires evidence of how she voted, to verify the vote was counted correctly and to be able to challenge the election if it was not. In the case of a write-in candidate, this means that the name entered may well be part of the voter’s retained evidence. The election authorities may also need to reveal the name in order to demonstrate that the write-in votes were correctly tallied. Thus such a system can allow a voter to enter some information of her choice into her vote, in a verifiable way. The challenge is to do this so as to minimise the leakage of information. In particular, as noted in [9], it is not acceptable for a receipt to provide evidence of having voted for a particular write-in candidate.

Elections allowing write-ins are rare outside the U.S., but they are common in the U.S. as part of the electoral landscape. Their implementation in the U.S. varies widely [12], since rules regarding write-ins are determined at state level. For example, in the 2008 Presidential election, some states did not allow write-in candidates; in others, write-in candidates had to register before the election; and

in others, any write-in is considered a legitimate candidate (subject to being eligible for office). It appears common practice that the totals for legitimate write-in candidates are reported as part of the election result. However, reporting of results for ‘unofficial’ write-in candidates may or may not occur. Write-ins are typically handled by a variety of technologies, including paper ballots and electronic voting systems.

There are therefore a range of possible legal requirements with respect to the results that must be reported. If a verifiable voting system is to support write-ins, then ideally it will not tally or report on more results than are required by the electoral rules. This will prevent unnecessary information leakage, as well as avoiding any unnecessary processing of votes. Hence it is desirable for such a system to provide flexibility with respect to the processing of the write-in votes.

The approach proposed in this paper also makes use of a ‘write-in’ option as an additional choice on a Prêt à Voter ballot form. The scheme we propose includes the write-in on the receipt and also on the bulletin board. This enables changes to any write-in image to be challenged by the receipt-holder. The inclusion of the write-in on the voter’s receipt introduces additional challenges that we address by incorporating the write-in into the mix. Election officials allocate write-in votes to candidates, and they are tallied homomorphically to protect voters’ privacy. Even if the receipt contains a recognisable write-in image which can be linked to a specific voter, the tallying procedure does not provide evidence of whether that vote was cast for that write-in candidate. We also introduce dummy receipts to allow a voter to vote for one write-in and obtain a receipt for another. This provides a stronger form of receipt-freeness with respect to write-ins than previous schemes.

Write-ins were first considered explicitly for cryptographic secure ballots in the Vector-Ballot scheme of [10], which uses a combination of mix networks and homomorphic encryption. It is noted in [2], that write-ins can potentially be incorporated into schemes such as Scratch&Vote [2], Prêt à Voter [20] and Punchscan [5], by including a ‘write-in’ option alongside the listed candidates, and incorporating a process such as the Vector-Ballot approach for handling the votes that are written in. This approach was required for the Takoma Park election [4] which was run using Scantegrity II [6]: in that election, voters ranked the candidates in order of preference, and so the system was extended to permit a write-in candidate in any position. This was achieved by incorporating a ‘write-in’ option and allowing it to be selected for any rank in the list. The associated write-in entry was scanned into the system, and processed by an ‘Election Resolution Manager’ component in order for it to be incorporated into the count. We discuss these schemes and compare them to ours in Section 5.

This paper is structured as follows: Section 2 describes the ballot form and how votes are cast, and discusses the nature of the receipt and the information it contains; Section 3 describes how the votes are processed through the two mixnets; Section 4 discusses the resulting scheme with respect to receipt-freeness; finally, Section 5 discusses other aspects of the scheme, and provides a comparison with other verifiable election schemes.

2 Incorporating Write-Ins into Prêt à Voter

Prêt à Voter uses cryptography in the construction of ballot forms, and in the processing of votes. The key features we require of the cryptosystem for the scheme presented in this paper are:

- probabilistic and semantically secure: that the same plaintext can be encrypted in many different ways, using different randomisation values, and that it should be computationally infeasible to tell whether two ciphertexts correspond to the same plaintext, without knowledge of the secret keys or randomisation involved;
- that a ciphertext can be re-encrypted without knowledge of the private key;
- that the cryptosystem is additively homomorphic: ciphertexts containing integers can be combined to a ciphertext containing their sum, without knowledge of the private key.

These properties are provided by the Paillier cryptosystem [17], and also by the ElGamal cryptosystem [7] using exponents to encode integers (since ElGamal itself is multiplicatively homomorphic).

In this paper we use the notation $E(m, r)$ to denote a message m encrypted under the election key with randomisation r . Re-encryption is denoted by changing the randomisation value, thus $E(m, r')$ is a re-encryption of $E(m, r)$ when r' is derived from r , and they both encrypt m . Decryption requires a threshold set of trusted parties, as discussed in [20].

2.1 The Ballot Form

The scheme incorporates write-ins by making an adaptation to the ballot form. Following the approach taken in [24], the ballot form is constructed with a randomised list of choices on the left hand side (and serial number at the top right) as illustrated in Figure 1. For plurality voting the possible permutations of the candidates could be simpler, as discussed in [22], but for this paper we will retain arbitrary candidate lists. The information capturing the candidate list is embedded in encrypted form in a code, illustrated here on the bottom of the ballot form. For each position on the ballot this gives an encryption of the associated candidate.

In common with other approaches, we introduce an additional ‘candidate’, ‘Write-In’, to the list of candidates on the left-hand side of the ballot, and include a space for a write-in candidate’s name on the right-hand side of the ballot. Recall that the left-hand side is to be destroyed, and the right hand side will be scanned into the system and recorded as the vote. Completing the ballot form is carried out as in previous versions of Prêt à Voter [20], with the added opportunity to write-in a candidate: the voter finds their choice on the randomised list of candidates (which now includes the additional choice ‘Write-In’) and marks an

	14
Daniel	
Ben	
Ali	
Write-In	
Cathy	
	<input type="checkbox"/> write-in name here <input type="checkbox"/>
	7xf2w-4j8yt

$E(\text{Daniel}, r_{14,d})$
 $E(\text{Ben}, r_{14,b})$
 $E(\text{Ali}, r_{14,a})$
 $E(\text{Write-in}, r_{14,w})$
 $E(\text{Cathy}, r_{14,c})$

Fig. 1. Prêt à Voter ballot form with write-ins

‘X’ in the corresponding box on the right-hand side; the voter may also write any name into the write-in box, which is on the right hand side.

2.2 Completing the Ballot Form

Figure 2 illustrates four ways of completing a valid ballot form. All except (d) are valid votes, though (c) also includes a decoy name (i.e. a write-in name who does not receive the vote).

- In vote (a), the voter has voted for Cathy, by selecting the box against Cathy’s name. No name has been written in the write-in box.
- In vote (b), the voter has chosen to cast a vote for a write-in candidate Bobbie. To do this, the voter selects the box against the candidate ‘Write-In’, and writes the name ‘Bobbie’ into the write-in box on the right-hand side of the ballot form,
- In vote (c), the voter has voted for Cathy. The name ‘Bobbie’ in the write-in box will not be counted as a vote for Bobbie, because the Write-In option was not selected.
- In vote (d), this is effectively a spoiled ballot: the write-in box was selected, but no name has been provided.

2.3 Dummy Ballot Forms

The scheme introduces the idea of dummy ballot forms, used to cast null votes. Here we first present the full scheme in which dummy ballots are provided to each voter. In Section 4.2 we discuss the alternative approach of making them optional for voters, and the consequences of not including them at all. Their purpose is to allow voters to obtain a receipt of any form with any write-in, while voting in any way they choose. A dummy ballot form is pictured in Figure 3. Following the construction of valid ballots, the code at the bottom of the form associates a (different) encryption of ‘null’ with each position.

Fig. 2. Four possible ways of completing the ballot form

Fig. 3. Prêt à Voter dummy ballot form

2.4 Casting a Vote

To cast their vote, voters are given a valid ballot form and also a dummy ballot form. There should be no relationship between the serial numbers of the two forms. They complete both, detach and destroy both left hand sides, scan both right hand sides into the system, and choose one receipt to retain. The receipt can be either that from the valid ballot form or from the dummy ballot form. The system must not know which receipt has been retained. One way to achieve this is to produce both receipts, and then require the voter to destroy one.

The right-hand side will either contain a write-in name, or not, as pictured in Figure 4

- If the right-hand side contains a write-in name, as pictured in Figure 4 (a), then either it is a vote for that candidate, or it is a vote for one of the listed candidates as illustrated in Figure 2, or else it is a null vote from a dummy ballot form. Without decrypting the code, and in the absence of the left-hand side, there is no way to determine which selection was made. Hence this right-hand side does not provide evidence that the written-in candidate actually received the vote.

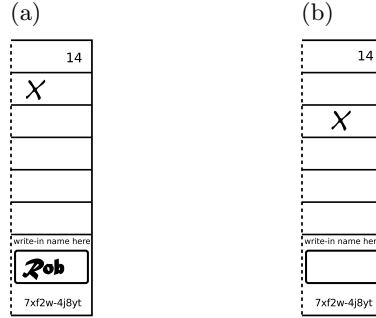


Fig. 4. Two possible forms of the right-hand side

- If the right hand side does not contain a written-in candidate, as pictured in Figure 4(b), then this vote is either a vote for one of the listed candidates, or a spoiled ballot, as illustrated in Figure 2, or else a null vote from a dummy ballot form. It does not indicate which of these possibilities actually holds.

3 Processing the Votes

Following recent versions of Prêt à Voter [20,21,19], we use a *re-encryption* mixnet [18,23,15] to process the votes.

3.1 Anonymising the Votes

A key feature of Prêt à Voter is the use of a public bulletin board to display the votes that are cast—the marked-up right hand sides of the ballot forms that the voters have submitted to the system. In this section we will consider that the record of any write-in will be recorded on the bulletin board as an image of the write-in box. The scheme works the same way if the write-ins are in text form.

When a vote v_i is scanned into the system, it is written to the bulletin board against the serial number on the ballot form, and the voter retains a (signed) receipt, which matches the bulletin board entry. What is recorded on the bulletin board and on the receipt is the onion associated with the voter’s selection, together with the readable image I_i of the entry in the write-in box. The onion for selection j on ballot form i is $E(C_j, r_{i,j})$: the selected candidate C_j in encrypted form with randomisation $r_{i,j}$.

At the end of polling, the votes are prepared for mixing by encrypting each serial number i with fixed randomiser 1, to obtain $E(i, 1)$. The results of this step are posted on the bulletin board and are easily verifiable. The entries to the mix are thus pairs of the form $(E(C_j, r_{i,j}), E(i, 1))$.

In principle the scheme could pass the scans I_i through the mix rather than pass the serial number i . However, a scan image is likely to be a bitmap of several kilobytes, and there will be practical problems encrypting such a large item and passing it through the mix. Using a serial number of a few tens of bits is significantly more efficient and less cumbersome.

The serial numbers are re-encrypted at each stage alongside the choice of candidate. At each stage of the mix each vote consists of an encrypted candidate and its associated encrypted serial number. The mix must not split the pairs at any stage. This can be achieved using the protocol of [15,14], or can be audited using mixnet techniques such as randomised partial checking [8]. Finally, the shuffled and re-encrypted votes are output from the mix, as another list of pairs of the form $E(C_j, r'_i), E(i, s'_i)$.

Tallying the Listed Candidates

At this stage, the encrypted candidates $E(C_j, r'_i)$ are decrypted in the usual way (by a threshold set of parties) to extract the candidates C_j that were selected. The serial numbers $E(i, s'_i)$ are not yet decrypted. The resulting list $\{(C_j, E(i, s'_i))\}$ is published. This process is illustrated in Figure 5.

After this stage, the votes have been decrypted, and so the election can be tallied. Any vote C_i for a listed candidate can be counted as a vote for that candidate. Any null vote is not counted against any candidate.

Tallying the Write-Ins

Any vote C_i for ‘Write-In’ cannot yet be counted, since the serial number linking to the image is encrypted and not yet available. It remains to tally the write-in candidates.

To obtain the write-in votes, we make use of a second mixnet. We use a homomorphic encryption scheme such as exponential ElGamal to encrypt the values 0 and 1 with a fixed randomiser, 1, to obtain known encryptions $E(0, 1)$ and $E(1, 1)$. This will enable us to obtain the aggregate values for each candidate. We take the list of results with their encrypted images $\{(C_j, E(i, s'_i))\}$ to generate a new list $\{(E(b_j, 1), E(i, s'_i))\}$, where each $E(b_j, 1)$ is the known encryption of 0 or 1, chosen as follows:

- if C_j is one of the listed candidates, then include $(E(0, 1), E(i, s'_i))$ in the input for the second mixnet;
- if C_j is ‘Null’, then include $(E(0, 1), E(i, s'_i))$ in the input for the second mixnet;
- if C_j is ‘Write-In’, then include $(E(1, 1), E(i, s'_i))$ in the input for the second mixnet

We include all the votes as input to the second mixnet: it is important to treat the decoy write-ins in the same way as the genuine write-in votes, otherwise the decoys will be exposed as such (because i would not eventually be revealed).

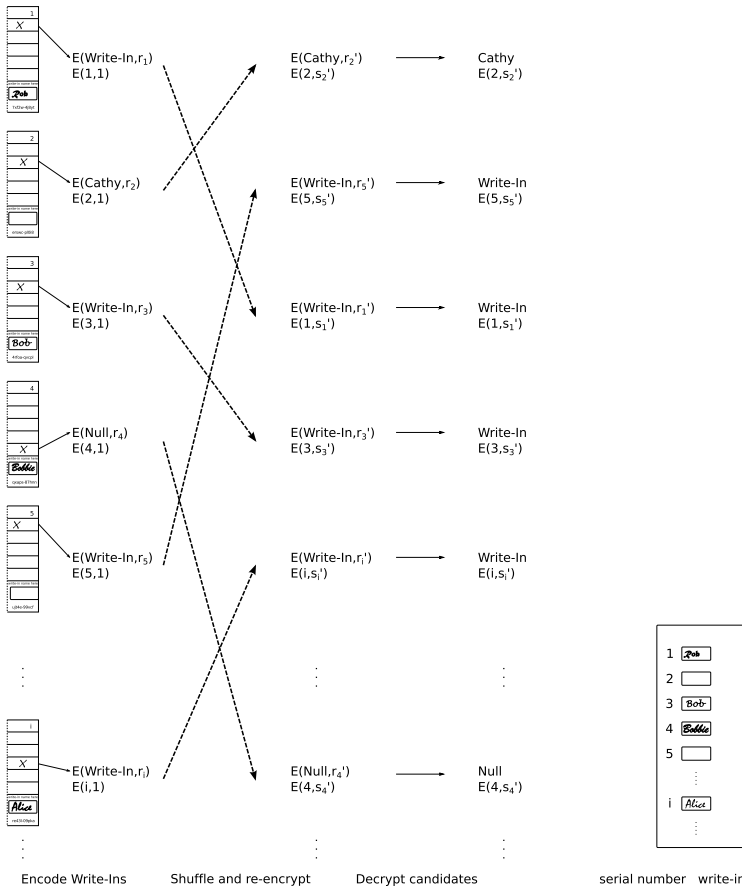


Fig. 5. Mixing the votes

The resulting list is run through the second mixnet, and is re-encrypted and shuffled. The resulting elements are of the form $(E(b_i, t_i), E(i, s_i''))$. The $E(i, s_i'')$ are now decrypted, to obtain all of the write-in entries. Note that the encrypted 0 or 1 values indicate whether or not those write-in entries corresponded to a ‘Write-In’ selection, i.e. whether that entry should be counted in the tally. This process is illustrated in Figure 6.

The write-in entries are examined, and assigned to particular candidates. This is the stage where election officials will need to make judgements as to the ‘voter’s intention’ in each case, and where procedural and legal challenges might take place. In due course, all of the write-ins are allocated to particular candidates. This process is independent of the processing of the ballots through the mixes, and can be carried out concurrently.

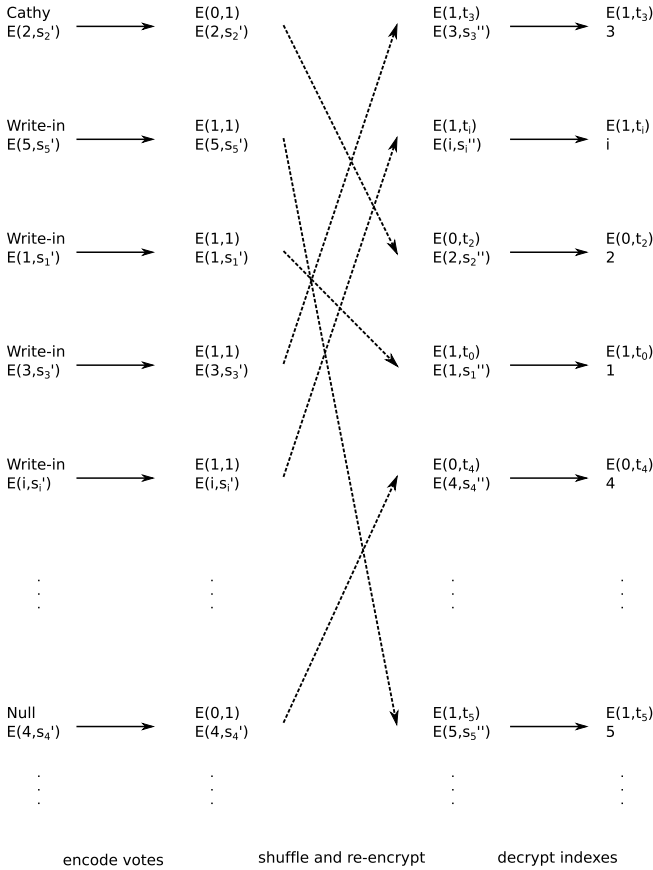


Fig. 6. Extracting the write-ins

The encrypted 0 or 1 values for each write-in candidate to be tallied can then be combined homomorphically: adding them together and then decrypting the result. This yields the total number of votes selected as ‘Write-In’ for that candidate, but it does not reveal for any particular vote whether that was a valid write-in, or a vote for one of the listed candidates or a null vote. This process is illustrated in Figure 7. Hence this process does not reveal whether the corresponding vote C_j was counted for the written-in candidate or not (except where this is evident from the result). This prevents the voter from obtaining a receipt for a vote for their write-in candidate.

3.2 Flexibility on Reporting Write-In Results

In practice different electoral regulations specify different rules for allowed write-ins and for reporting on write-ins, giving rise to a whole range of situations that

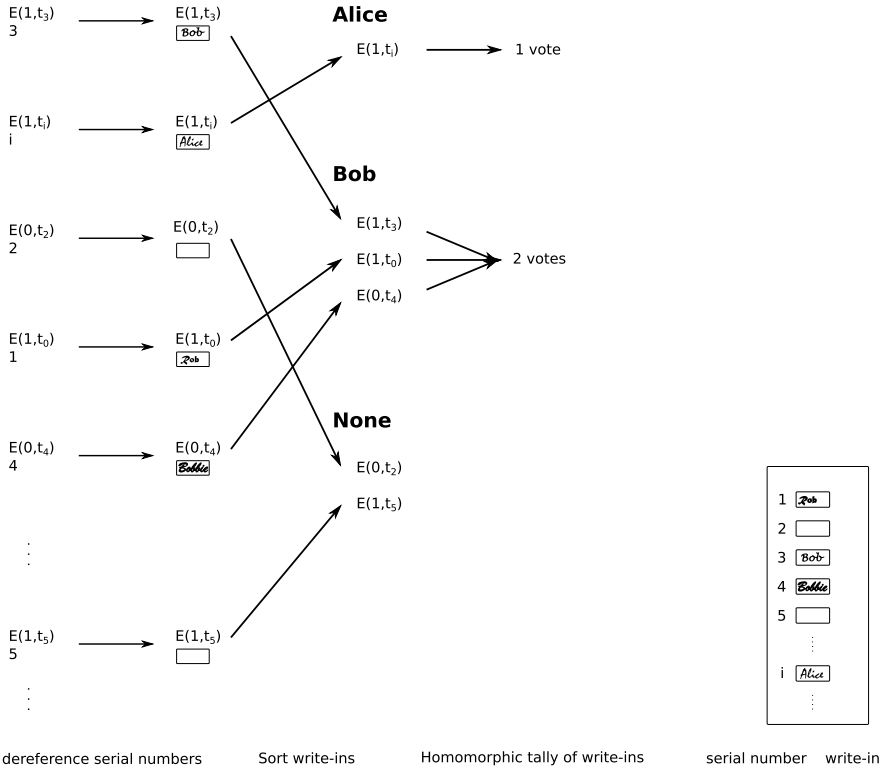


Fig. 7. Tallying the write-ins

any election system might need to deal with, from reporting tallies for all written-in names, to only reporting on tallies of ‘official’ write-in candidates (who have registered with the electoral authorities in some way). The output of the first mix (Figure 5) indicates the total number of write-ins; and the scans of the write-ins on the bulletin board, once assigned to candidates, give an upper bound for the number of votes received by each write-in candidate. If it is apparent from these intermediate numbers that the write-in votes cannot make a difference to the result of the election, it may be allowable to report on the result even before the write-ins are tallied.

The write-ins on the bulletin board can be interpreted and sorted by candidate, and only those to be reported need to be tallied by means of the second mix and homomorphic tallying. Thus the stages of the tallying process can be tailored to the local regulations regarding write-ins.

4 Receipt-Freeness

Informally, receipt-freeness is the property that a voter cannot obtain evidence of how they voted. Benaloh and Tuinstra [3] introduced the issue of receipt-freeness,

and considered it as the property that does not allow a voter “to prove that a vote was cast in a particular way”.

Okamoto [16] gives a formal definition of receipt-freeness, formalising the property as the ability of the voter to cast a vote v_i so that a coercer cannot tell that their preferred vote v_i^* has not been cast. In the case where a receipt is issued, this means that the receipt obtained by the voter following a vote v_i is consistent with a vote for v_i^* . For the purposes of this paper, we will consider what the coercer can learn from the information contained in the receipt retained by the voter.

Even at this level of abstraction, an analysis is useful in the context of write-ins since the asymmetry between listed candidates and write-in candidates already introduces some interesting considerations. A full security analysis for receipt-freeness will require more detailed modelling of the system: the Universal Composability approach [13] is a good candidate to provide an appropriate framework, as it provides a rigorous model for coercion, incorporates the modelling of the system, and allows for the information revealed by the posting of receipts on the bulletin board and the tallying of the election. This is the subject of ongoing research.

We adapt Okamoto’s definition of receipt-freeness to our current setting, referring to receipts explicitly, as follows:

Definition 1 (After Okamoto). *A voting scheme with receipts is receipt-free if a coercer who requires the voter to vote for candidate c_i^* , and will accept a receipt r_i^* , cannot tell that the voter’s preferred candidate c_i did not receive the vote.*

In other words, any receipt that a coercer will accept is consistent with a vote for any candidate.

Classical Prêt à Voter is receipt-free in this sense: the receipt that the voter obtains could correspond to a vote for any of the candidates, and so it does not provide any evidence of which candidate received the vote, or any useful information about the vote.

In the scheme incorporating write-ins presented in Sections 2 and 3, any receipt retained by a voter is consistent with any valid vote that she may have cast, including a valid vote for any write-in candidate. Since a dummy receipt can be of any form at all, and bears no relationship to the valid vote cast, any receipt a voter might retain is consistent with any valid vote. The scheme thus meets the characterisation of *receipt-freeness* given in Definition 1 above.

4.1 The Scheme without Dummy Ballots

If we restrict the scheme to using only valid ballots, and not dummy ballots, then the receipt-freeness properties are weakened with respect to write-in candidates, and receipts do leak some information:

- A receipt of a valid ballot, with a particular write-in, is not consistent with a vote for a different write-in candidate. It is consistent with a vote for the written-in candidate, and for any of the listed candidates.

- A receipt of a valid ballot with no write-in is not consistent with a vote for any write-in. It is consistent with a vote for any of the candidates, and for a spoiled ballot (i.e. a vote for a write-in but no candidate written-in).

Hence the system without the use of dummy ballots does *not* provide receipt-freeness in the case of write-ins. While any receipt is consistent with any vote for a listed candidate, it is consistent with a vote for a write-in candidate only if that name is written in the write-in box. If receipt r_i^* does not contain a write-in name, or contains a write-in name other than wi_i , then a coercer *can* tell that a voter's preferred (write-in) candidate wi_i did not receive the vote; and thus Definition [1](#) does not hold.

Observe that the weaker form of Definition [1](#) below does hold, where the voter's preferred candidate c_i is a listed candidate:

Definition 2 (Receipt-freeness for listed candidates). *A voting scheme with receipts is receipt-free for listed candidates if, for any listed candidate c_i , a coercer who requires the voter to vote for candidate c_i^* , and will accept a receipt r_i^* , cannot tell that the voter's preferred listed candidate c_i did not receive the vote.*

The voter may vote for any listed candidate while remaining consistent with any receipt a coercer might require. However, this definition is too weak to give full receipt-freeness: the voter does not have the required guarantees in the case where she wishes to vote for a write-in candidate.

4.2 Optional Dummy Ballots

Rather than give both a valid and a dummy ballot form to each voter, a suggested approach is to offer the voter the *option* of whether they wish to take a dummy alongside a valid form, rather than making it compulsory. An alternative approach could offer voters the opportunity to take as many dummy ballots as they wished.

By making the choice optional, voters still have the ability to obtain a dummy receipt if required, thus retaining receipt-freeness of the system.

In this case it is essential that a coercer should not know whether or not a voter opted to take the dummy ballot. Otherwise the additional knowledge that a voter did not select a dummy reduces receipt-freeness to the case where dummy ballots are not used. This means that the procedure for offering the voter the choice, and providing the voter with the appropriate ballot form(s), must be carefully designed to prevent information leaking to the coercer.

Dummy votes do not contribute to the final election result, and are provided purely to provide receipt-freeness. Since dummy ballots are processed in exactly the same way as valid ballots, their inclusion introduces a processing overhead, and reducing the number of dummy ballots passed into the system will improve efficiency. Making dummy ballots optional for voters provides a good way of achieving this.

5 Discussion

5.1 Human Factors

The inclusion of a write-in option on the ballot form introduces an inherent asymmetry between candidates: write-in candidates are handled differently to listed candidates. While we see that any receipt is compatible with any vote, in practice it may be that voters will use the ballot forms in particular ways. For example it seems plausible that many voters will only complete the write-in box if they are voting for that write-in candidate, and that they will otherwise leave it blank. Such a pattern of behaviour would impact on the information given by a receipt: a receipt including a write-in name might in practice indicate a higher likelihood of a vote for that candidate than a receipt without that name written-in. Empirical investigations would be necessary to gain an understanding of the bias introduced by voter practise.

5.2 Comparison with Other Schemes

Few proposals for secure election systems explicitly address write-in ballots. For example, the Helios online election system [1] does not address the issue of write-ins explicitly, and the current implementation does not provide for them. The JCJ scheme of [9] excludes write-ins, because of the concern that an attacker could coerce voters to introduce specific strings into the write-in box and check that they have done so. Our scheme minimises the impact of such coercion, since the presence of a write-in on the Prêt à Voter receipt is not evidence of a vote for that write-in candidate.

In this section we will consider two schemes which do allow write-ins: Vector Ballots, and Scantegrity II. We will consider them with respect to the various aspects discussed above.

Vector Ballots

The Vector Ballot approach described [11] is to our knowledge the only cryptographic e-voting protocol designed specifically to support write-in ballots. It is a purely electronic system, intended for internet voting. A vector ballot accepts both write-in candidates and votes for listed candidates. A completed ballot is a vector with three components: an encrypted flag indicating whether the vote is for a write-in or a listed candidate; a ciphertext possibly containing a listed candidate; and a ciphertext possibly containing a write-in. Correctly completed ballots are either

1. an encrypted flag value of 0, an encryption of a selected candidate, and an encryption of 0 for the write-in; or
2. an encrypted flag value of 1, an encryption of 0 for the selected candidate, and an encryption of the write-in candidate name.

The non-write-in ballots can then be tallied using homomorphic encryption. The write-ins cannot be tallied homomorphically since they are not predetermined; they must be extracted and revealed individually in order to enable tallying. This is achieved by a ‘shrink and mix’ procedure: the sequence of ballots is divided up into batches, and then for each batch, if *any* of the write-in flags are set, then the whole batch is included as input to the mix. However, any batch with no write-ins will not be input to the mix. Hence all write-ins, and a number of non-write-ins, will be input to the mix, but the list of ballots is reduced. The motivation for this is to improve efficiency: mixes are computationally expensive, and since only small proportions of voters typically opt for write-ins, many of the ballots can be excluded from the mix while still masking which of the input ballots are write-ins. The output of the mix can then be decrypted, the null write-ins discarded, and the genuine write-ins tallied.

With regards to receipt-freeness, the only information provided by the system is whether the ballot is input into the mix. If it is input then the ballot could be for a listed candidate or a write-in. However, if it is not included in the input then it is clearly not for a write-in candidate. Hence it is possible (indeed, likely) for voters voting for a listed candidate to obtain evidence that they did not vote for any write-in candidate. However, it is possible for all votes to be put into the mix by considering all the votes as one batch, and not applying the ‘shrink’ stage of the process. Hence if necessary the scheme can provide receipt-freeness of not having voted for a write-in, though at the cost of efficiency of the tally.

The scheme does not allow decoy write-ins. Although voters’ receipts (i.e. what is on the bulletin board as received votes) do not expose write-in candidate entries, they are exposed in the output from the mix, and all such votes will have been cast as write-ins. Since these can in principle contain arbitrary strings, voters can be subjected to the forced abstention attack by being coerced into entering such strings. The appearance of such a string in the mix output provides evidence that the voter did indeed waste their vote.

This last property contrasts with our approach, where the published material does not indicate whether the string was selected as a write-in. If the authorities do not tally the results for such a string, then its appearance does not provide evidence of a wasted vote.

Scantegrity II

The description of Scantegrity II in [6] does not explicitly discuss write-ins, but they were incorporated into the system to meet the election requirements of the Takoma Park municipal election which was run using Scantegrity II. The system takes a different approach to the systems discussed above, in that the voters do not retain any receipt of what they have written in, so they are not able to demonstrate whether they have or have not written-in any particular candidate. In this sense, the Scantegrity II approach provides a stronger form of receipt-freeness than the others with respect to write-ins. In that system, the final tally for ‘Write-In’ is verifiable, but a level of trust is required in the election authority, since a malicious authority

could change the write-in images [4]. If necessary, additional procedures could be introduced to verify the write-ins. This contrasts with our scheme, which allows voters to confirm that the write-in image they provided has not been changed by virtue of the receipt and the published election data.

6 Summary

We have presented an extension of Prêt à Voter to include a mechanism for allowing write-in candidates. The approach provides the voter with a receipt which can contain an image of the (hand-written) write-in but still provides receipt-freeness, in the sense that the voter's receipt is not evidence of the voter's vote. The scheme provides maximum flexibility when tallying, allowing write-ins to be assessed and assigned against write-in candidate names by election officials before they are tallied, and tallying only when necessary according to local requirements. Homomorphic tallying allows write-ins to be tallied without revealing which particular votes were cast for them, providing secrecy of the ballot. We have discussed various aspects of the property of receipt-freeness with respect to the information contained in the receipt, and seen the need for dummy or null ballots to provide receipt-freeness for write-ins. We believe that the approach taken in this paper and developed for Prêt à Voter will be more generally applicable to other voting schemes in which a voter obtains a receipt of how they marked their ballot form.

Acknowledgements. We are grateful to Ron Rivest, Peter Ryan, and Emily Shen for discussions and comments on these ideas. This work was funded by EPSRC under grant EP/G025797/1, and was conducted while James Heather was a Royal Academy of Engineering/Leverhulme Trust Senior Research Fellow.

References

1. Adida, B.: Helios: Web-based open-audit voting. In: van Oorschot, P.C. (ed.) USENIX Security Symposium, pp. 335–348. USENIX Association (2008)
2. Adida, B., Rivest, R.L.: Scratch & vote: self-contained paper-based cryptographic voting. In: Juels, A., Winslett, M. (eds.) WPES, pp. 29–40. ACM (2006)
3. Benaloh, J.C., Tuinstra, D.: Receipt-free secret-ballot elections (extended abstract). In: STOC, pp. 544–553 (1994)
4. Carback, R., Chaum, D., Clark, J., Conway, J., Essex, A., Herrnson, P.S., Mayberry, T., Popoveniuc, S., Rivest, R.L., Shen, E., Sherman, A.T., Vora, P.L.: Scantegrity II municipal election at Takoma Park: The first E2E binding governmental election with ballot privacy. In: Proceedings of the 19th USENIX Security Symposium (2010)
5. Chaum, D.: Punchscan, <http://www.punchscan.org> (viewed on November 12, 2010)
6. Chaum, D., Carback, R., Clark, J., Essex, A., Popoveniuc, S., Rivest, R.L., Ryan, P.Y.A., Shen, E., Sherman, A.T., Vora, P.L.: Scantegrity II: end-to-end verifiability by voters of optical scan elections through confirmation codes. IEEE Transactions on Information Forensics and Security 4(4), 611–627 (2009)

7. El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory* 31(4), 469–472 (1985)
8. Jakobsson, M., Juels, A., Rivest, R.L.: Making mix nets robust for electronic voting by randomized partial checking. In: Boneh, D. (ed.) *USENIX Security Symposium*, pp. 339–353. *USENIX* (2002)
9. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: Atluri, V., et al. (eds.) *WPES*, pp. 61–70. *ACM* (2005)
10. 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)
11. Kiayias, A., Yung, M.: The Vector-Ballot Approach for Online Voting Procedures. In: Chaum, D., Jakobsson, M., Rivest, R.L., Ryan, P.Y.A., Benaloh, J., Kutyłowski, M., Adida, B. (eds.) *Towards Trustworthy Elections*. LNCS, vol. 6000, pp. 155–174. Springer, Heidelberg (2010)
12. mfooster.com. 2008 Presidential Election Write-In Rules, mfooster.com/misc/write-in-rules-2008.html (viewed on November 15, 2010)
13. 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)
14. Neff, A.D.: Verifiable mixing (shuffling) of ElGamal pairs (2004), <http://web.archive.org/web/20041212174250/www.votehere.net/documents.html> (last accessed March 1, 2011)
15. 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)
16. 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)
17. 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)
18. Park, C., Itoh, K., Kurosawa, K.: Efficient Anonymous Channel and All/Nothing Election Scheme. In: Helleseeth, T. (ed.) *EUROCRYPT 1993*. LNCS, vol. 765, pp. 248–259. Springer, Heidelberg (1994)
19. Ryan, P.Y.A.: Prêt à Voter with Paillier encryption. Technical Report CS-TR-965, University of Newcastle (2006)
20. Ryan, P.Y.A., Bismark, D., Heather, J., Schneider, S., Xia, Z.: Prêt à Voter: a voter-verifiable voting system. *IEEE Transactions on Information Forensics and Security* 4(4), 662–673 (2009)
21. 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)
22. Ryan, P.Y.A., Teague, V.: Ballot permutations in Prêt à Voter. In: *USENIX/ACCURATE Electronic Voting Technology Workshop* (2009)
23. 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)
24. Xia, Z., Culnane, C., Heather, J., Jonker, H., Ryan, P.Y.A., Schneider, S., Srinivasan, S.: Versatile Prêt à Voter: Handling Multiple Election Methods with a Unified Interface. In: Gong, G., Gupta, K.C. (eds.) *INDOCRYPT 2010*. LNCS, vol. 6498, pp. 98–114. Springer, Heidelberg (2010)

Trivitas: Voters Directly Verifying Votes

Sergiu Bursuc, Gurchetan S. Grewal, and Mark D. Ryan

School of Computer Science, University of Birmingham, UK
{s.bursuc,m.d.ryan}@cs.bham.ac.uk, research@gurchetan.com

Abstract. Individual verifiability is the ability of an electronic voting system to convince a voter that his vote has been correctly counted in the tally. Unfortunately, in most electronic voting systems the proofs for individual verifiability are non-intuitive and, moreover, need trusted devices to be checked. Based on the remote voting system JCJ/Civitas, we propose Trivitas, a protocol that achieves direct and end-to-end individual verifiability, while at the same time preserving coercion-resistance.

Our technical contributions rely on two main ideas, both related to the notion of credentials already present in JCJ/Civitas. Firstly, we propose the use of trial credentials, as a way to track and audit the handling of a ballot from one end of the election system to the other end, without increased complexity on the voter end. Secondly, due to indistinguishability of credentials from random values, we observe that the association between any credential and its corresponding vote can be made public at the end of the election process, without compromising coercion-resistance. The voter has more intuitive and direct evidence that her intended vote has not been changed and will be counted in the final tally.

Keywords: Electronic voting, Individual verifiability, Trial votes, Intuitive verifiability.

1 Introduction

The concept of *election outcome verifiability* has emerged as a vital ingredient in electronic voting systems. This has arisen partly because some implementations have been shown vulnerable to outcome manipulation, e.g. [8,5]. Another reason is that, in contrast with electronic banking and social networking, it is not easy to put mistakes right if they are uncovered after the results of the election have been declared. A third reason is that it is notoriously difficult to prove that the *software* which might be running behind the scenes has the expected properties; it is more practical to verify the results produced by the software, than the software itself. For this reason, election outcome verifiability is sometimes referred to as *software independence*.

Election verifiability in presence of a bulletin board may be conveniently split in three notions [19]:

Individual Verifiability [9,21,10,2,3,4,1]. To any voter, individual verifiability should offer the possibility to verify that her cast ballot has been correctly

recorded and tallied by the system. Ideally, the voter should also have an assurance that her cast ballot correctly encodes her cast vote. In some systems this additional assurance is offered by an option to audit a ballot [3,4,1] or a ballot form [10] before casting a vote.

Universal Verifiability [17,20,14,15]. To any external observer, universal verifiability should offer the possibility to verify that all recorded votes have been correctly tallied.

Eligibility Verifiability [18,11,19]. To any external observer, eligibility verifiability should offer the possibility to verify that the set of tallied votes corresponds to votes cast by eligible voters. Eligibility may be enforced at any stage in the system: either during the casting of a vote [10], or during its recording [1], or during its tallying [18,11]. Accordingly, eligibility verifiability will pertain to the corresponding stage.

One problem with the way *individual verifiability* is usually achieved is that the voter does not see her vote at the time she visits the bulletin board. This is the case in Helios [1], Pret-a-Voter [10], JCJ/Civitas [18,11], and others. The reason is in order to achieve the property of coercion-resistance, which asserts that the voter shouldn't be able to prove to a potential coercer how she voted. Therefore, individual verifiability is achieved by indirect means; the voter can check that the encrypted ballot is present, and has some other evidence (perhaps based on auditing) that the encrypted ballot really represents her vote. Moreover, after the ballots are anonymized, the voter loses track even of her encrypted ballot.

Voters are likely to find this indirect achievement of individual verifiability unsatisfactory. This feeling has arisen in the focus groups that were held as part of the EPSRC project *Trustworthy Voting Systems* [23]. Four hour-long managed discussions among groups of about 10 citizens were arranged by a professional facilitator, with the aim of soliciting people's opinions about Pret-a-Voter. In at least two of the discussions, participants questioned the worthwhileness of checking the presence of their ballot on the bulletin board, given that they did not have any direct evidence that the ballot really contained their vote. The issue has also been mentioned by Adida and Neff [2], where the requirement that verifiability should be *direct* and *end to end* has been highlighted.

Our Contribution. We introduce Trivitas, an adaptation of JCJ/Civitas. We show how the credentials of JCJ/Civitas can be adapted to improve individual verifiability. In particular, we show how voters can see their own vote in plaintext, making the verification experience more direct, and more intuitive.

Our first idea is the notion of trial votes. A trial vote is a vote that is cast along with real votes, but will not be counted. It will be decrypted and exposed along the way, in a way that is traceable by the voter that cast it. Since most of the system components are not able to distinguish trial and real votes, it gives confidence in the correct handling of all votes. This is an extension of the ideas of auditing in [10,11], with the crucial difference that the auditing process is performed not only in the phase of casting a ballot, but is spread throughout

the whole election process. In other words, a trial credential will function as a marker whose sign is that the handling of this ballot should be made transparent, by e.g. decrypting and publishing its contents at every stage. There are a few technical difficulties with that, because trial ballots can fulfill their role only as long as they are not identified as such by possibly corrupted agents in the voting system. To avoid this problem, we make use of the fact that the decryption key is distributed among a set of trustees and propose to decrypt trial ballots by running a decryption mix [9,10] among the trustees.

Our second idea is based on the following observation: real credentials are indistinguishable for anyone (except for the voter) from trial credentials and fake credentials (an element that JCJ/Civitas introduces to enable coercion-resistance). Therefore, without compromising coercion-resistance, for each ballot (be it real, trial or fake) we can publish after anonymization (done by a re-encryption mixnet [17]) its corresponding credential and the decrypted vote. This allows a voter to verify that all its votes have made it into the final tally: its real vote, its trial vote and its possible fake votes. There is again a technical difficulty, related to eligibility verifiability and coercion-resistance: trial votes and fake votes have to be eliminated from the final tally in a publicly verifiable way, hence a coercer could observe that the credential obtained from the voter is fake. To avoid this problem we make use again of a decryption mix run by the trustees: there is no way to link the decrypted contents of a ballot to ballots that will be eliminated to enforce eligibility.

Outline of the Paper. In section 2 we describe the cryptographic primitives used in JCJ/Civitas and in section 3 we review its design and its solutions for election verifiability. Then, we make specific our critique of individual verifiability, that can be extended to systems like Pret a Voter [10] and Helios [1]. In section 4, we describe our proposal. In section 5, we show initial ideas about how trial credentials could be used to improve universal verifiability and recoverability. Finally, in section 6 we argue that changing JCJ/Civitas in the way that we propose does not compromise the coercion-resistance guarantees of the original system, and we give a hint of how the proof of [18] could be adapted to prove coercion-resistance for the new system.

2 Cryptographic Primitives

JCJ/Civitas relies on the following cryptographic primitives. We do not detail the structure of zero-knowledge proofs, because it is not relevant for our purposes.

Distributed El-Gamal [7]. The encryption scheme being used is El-Gamal over a multiplicative group of integers modulo a prime $p = 2kq + 1$, where q is also prime. The plaintext space and the ciphertext space \mathcal{M} is the order q subgroup of \mathbb{Z}_p^* (actually, some encoding has to be done when one wants the plaintext space to be \mathbb{Z}_q , but we can ignore such details here). Let g be the generator of \mathcal{M} . Then, a private key is a number $x \in \mathbb{Z}_q^*$ and the corresponding

public key is $k = g^x \bmod p$. The encryption of a message m with a public key k is a pair $(g^r \bmod p, m \cdot k^r \bmod p)$ where r is a fresh random number in \mathbb{Z}_q^* . To decrypt a ciphertext (a, b) the holder of the private key x computes $\frac{b}{a^x} \bmod p$.

JCJ/Civitas distributes the secret key (relying on e.g. [22]) among a set of trustees T_1, \dots, T_n . In that case, the private key is split as $x = x_1 + \dots + x_n$ and each of T_i holds a secret share x_i . To decrypt a ciphertext (a, b) , each of T_i publishes a partial decryption share $a_i = a^{x_i} \bmod p$. The final decryption can then be publicly computed as $\frac{b}{a_1 \dots a_n}$.

The El-Gamal encryption of a plaintext m with a public key k and random r will be denoted in the following by $\{m\}_k^r$, or simply by $\{m\}_k$ when r is not important or is clear from the context. The private part of a public key k will be denoted by $\text{priv}(k)$. The decryption of a ciphertext m with a private key x will be denoted by $\text{dec}(m, x)$, or simply by $\text{dec}(m)$ when the key is clear from the context.

Decryption Proof. Along with partial decryption shares a_i the holders of private key shares can send a zero-knowledge proof (equality of discrete logarithms) of the fact that they have used the correct key share to construct a_i . This allows any observer to check that the final result of the decryption is correctly computed, i.e. that decryption of $\{m\}_k$ is performed with the key $\text{priv}(k)$ and gives the result m .

Re-encryption and Mix Nets. Given a ciphertext (a, b) constructed using the public key k any party can compute another ciphertext (a', b') such that (a', b') encrypts the same plaintext as (a, b) using the same key k : choose a random $r \in \mathbb{Z}_q^*$ and let $(a', b') = (a \cdot g^r \bmod p, b \cdot k^r \bmod p)$. We will denote the re-encryption of a ciphertext m with a random r by $\text{renc}(m, r)$.

A re-encryption mix net \mathcal{M} takes as input a sequence of ciphertexts $\mathcal{S} = m_1, \dots, m_k$ and outputs a sequence of ciphertexts $\mathcal{S}' = m'_1, \dots, m'_k$ that is a re-encryption mix of \mathcal{S} . That is, \mathcal{S}' is formed by re-encryption of elements in a permutation of \mathcal{S} . Formally, there is a permutation σ of $\{1, \dots, k\}$ and a sequence of randoms r_1, \dots, r_k such that $m'_1 = \text{renc}(m_{\sigma(1)}, r_1), \dots, m'_k = \text{renc}(m_{\sigma(k)}, r_k)$. Moreover, if at least one member of \mathcal{M} is trusted not to be controlled by an intruder, he is unable to discover the permutation σ .

Mix Proof, e.g. [17]. Given two sequences of ciphertexts $\mathcal{S} = m_1, \dots, m_k$ and $\mathcal{S}' = m'_1, \dots, m'_k$, a zero-knowledge mix proof shows that \mathcal{S}' is a correct re-encryption mix of \mathcal{S} , but without revealing (a non-negligible part of) σ .

Plaintext Equivalence Test [16]. Consider two ciphertexts (a_1, b_1) and respectively (a_2, b_2) , encrypted with the same key k , whose plaintexts are t_1 and respectively t_2 . Assume that the private part $\text{priv}(k)$ is distributed among T_1, \dots, T_n . Then T_1, \dots, T_n can run a protocol to determine if $t_1 = t_2$ without them being able to learn t_1 or t_2 : roughly, they compute $(a, b) = (\frac{a_1}{a_2}, \frac{b_1}{b_2})$ and perform a distributed decryption of (a, b) ; if the result of the decryption is 1, then $t_1 = t_2$; otherwise, $t_1 \neq t_2$.

For two ciphertexts m_1 and m_2 , we will denote by $\text{PET}(m_1, m_2) = \text{true}$ if the plaintext equivalence test holds for m_1 and m_2 .

PET Proof [16]. Decryption proofs for the distributed decryption performed in a plaintext equivalence test can be used to attest that the test has been correctly performed, i.e. that $\text{PET}(m_1, m_2) = \text{true}$ if and only if $\text{dec}(m_1, \text{priv}(k)) = \text{dec}(m_2, \text{priv}(k))$.

3 Individual Verifiability in JCJ/Civitas

3.1 Overview of JCJ/Civitas

The main idea in JCJ/Civitas is the notion of credentials (with a private and a public part), that allow eligible voters to authenticate their ballots. To allow coercion-resistance, JCJ/Civitas distributes credential generation among a set of parties called registrars. It is assumed that at least one of the registrars will not be corrupted by the coercer and that the voter can communicate with this registrar using an untappable channel. To evade coercion, the voter has the ability to generate a fake credential, that is indistinguishable from a real one for the coercer. The participants of the protocol are

\mathcal{R} - the set of registrars, whose role is to authenticate eligible voters and help generate their credentials.

\mathcal{T} - the set of trustees, whose role is to generate and publish the public key of the election. Each of them holds a secret share of the corresponding private key, that will be used for distributed decryption and plaintext equivalence tests.

$\mathcal{V}_1, \dots, \mathcal{V}_n$ - the set of eligible voters.

\mathcal{M} - a re-encryption mix net, whose role is to anonymize the set of cast ballots before verification of their eligibility and their decryption.

\mathcal{B} - the bulletin board, whose role is to record the manipulation of ballots at all stages of the election, from their recording to their tallying. It also records proofs of correct ballot handling submitted by \mathcal{R} , \mathcal{T} and \mathcal{M} , that can be checked by external auditors.

A coercer may control some of $\mathcal{V}_1, \dots, \mathcal{V}_n$, some of \mathcal{R} , some of \mathcal{T} and some of \mathcal{M} but not all. To achieve coercion-resistance, at least one of \mathcal{R} , one of \mathcal{T} and one of \mathcal{M} has to be outside the control of the coercer. A summary of the protocol is as follows, complemented by solid lines in figures 1 and 2. There are three phases: registration, voting and tabulation.

Registration. Election starts with trustees generating the public key KT of the election in a distributed manner, such that no minority of trustees can recover the private key $\text{priv}(\text{KT})$ [22] and the decryption is distributed [7]. The public part of the key is published on the bulletin board. By running a separate protocol with each of the registrars, the voter V_i obtains the private part c_i of her credential, together with a non-transferable proof P_i of the fact that the public part $\{c_i\}_{\text{KT}}$, that is published in the electoral roll ER , correctly encodes the private part.

Voting. The ballot contains the encryption of the private credential c_i (with the key KT and with a different random than in ER) and the encryption of the intended vote v_i (with the key KT). To prevent the re-use of the same credential by a party that does not hold the private part, the ballot contains additionally a proof P_{cv} of the fact that its creator knows both c_i and v_i . Additionally, P_{corr} proves that v_i is a valid vote, according to the specification decided by election authorities.

Tabulation. Before tabulation starts, the proofs of cast ballots are verified and ballots with invalid proofs are discarded. The valid ballots and the electoral roll are then sent to a re-encryption mix net for anonymization. Credentials from anonymized ballots are compared to credentials from the anonymized electoral roll, to ensure that votes to be counted are cast by eligible voters only. If multiple ballots are submitted with the same credentials, only one copy is kept according to a predefined policy, e.g. only the last vote counts. Duplicate elimination is done by plaintext equivalence tests and can be performed either before, or after the mix. Finally, the decided set of countable votes is decrypted, and the corresponding decryption proofs are posted on the bulletin board.

3.2 Election Verifiability

Universal verifiability and **eligibility verifiability** are achieved by proofs posted on the bulletin board:

- Mix proofs allow auditors \mathcal{A} to verify that all ballot coming out of the mix net \mathcal{M} (i.e. belonging to the set MixedBallots) corresponds to a recorded ballot (i.e. belonging to the set CastBallots), and also that no recorded ballot has been discarded.
- PET proofs and proofs P_{cv} allow \mathcal{A} to verify that only votes coming from eligible voters are kept on the bulletin board for final decryption (i.e. in the set CountableVotes).
- Decryption proofs allow \mathcal{A} to verify that all countable votes have been correctly decrypted.

Individual verifiability is achieved as follows:

1. \mathcal{V} must trust her voting machine that her cast ballot correctly encodes her vote.
2. \mathcal{V} can check the bulletin board to see that the cast ballot has been correctly recorded.
3. Universal verifiability of mix nets ensures that all the recorded votes are correctly mixed, and therefore the vote cast by \mathcal{V} is included in the set of mixed ballots.
4. Universal verifiability of PETs ensures that at least one copy of \mathcal{V} 's ballot is kept after the elimination of duplicates. Moreover, the proof P_i that the voter obtains during registration ensures that her private credential corresponds to a public credential in the electoral roll ER . Universal verifiability of mix nets ensures that a re-encryption of her public credential is also present in

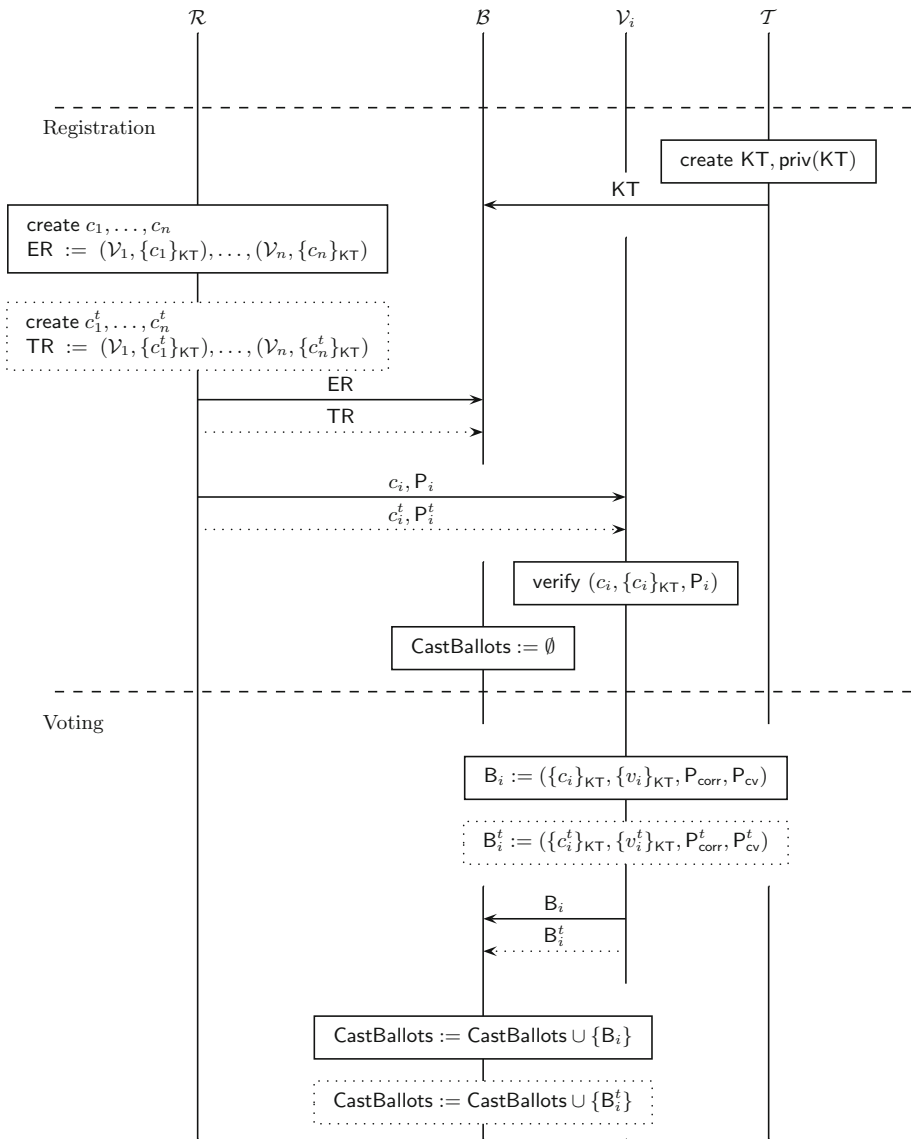


Fig. 1. JCJ/Civitas (solid lines) and additions of Trivitas (dotted lines): Registration and Voting phases

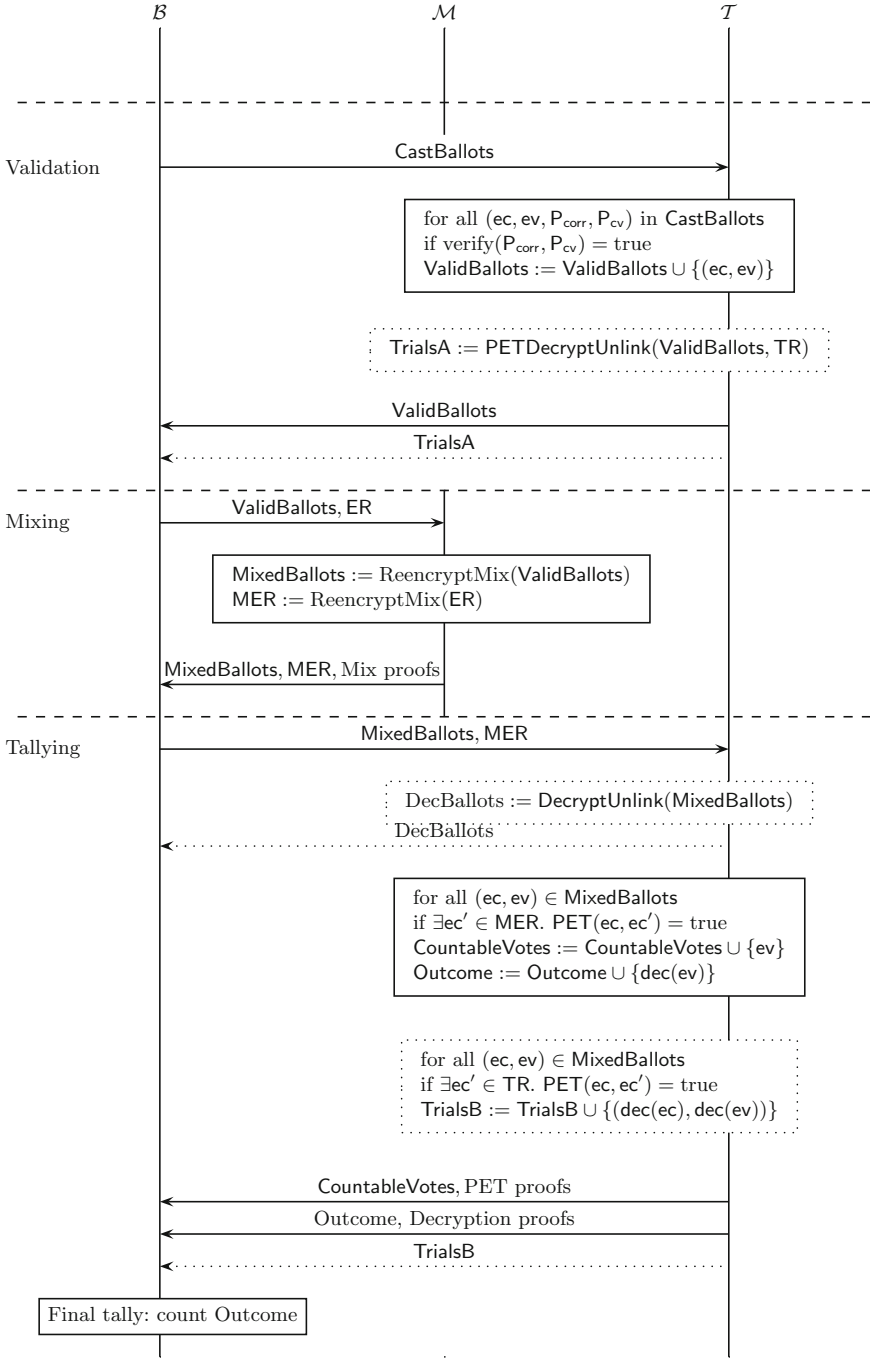


Fig. 2. JCJ/Civitas (solid lines) and additions of Trivitas (dotted lines): Tabulation phase

the anonymized electoral roll MER. Universal verifiability of PETs ensures that valid ballots are not eliminated when credentials are checked against the electoral roll MER. Altogether, these give to \mathcal{V} an assurance that her ballot is identified as coming from an eligible voter and is not eliminated during the enforcement of eligibility.

5. Finally, universal verifiability of distributed decryption ensures that \mathcal{V} 's ballot is correctly decrypted and tallied.

Some systems, e.g Pret a Voter [10] and Helios [1], improve the first point with a *cut-and-choose* mechanism, that allows the voter to audit a ballot before casting a vote.

Our critique of individual verifiability in JCJ/Civitas, Pret a Voter and Helios refers more generally to systems that rely on universal verifiability to achieve end-to-end individual verifiability. The points 3-5 above require complex mathematical operations and the corresponding verification algorithms must be run on trusted devices. Moreover, even if the corresponding zero-knowledge proofs are rigorously tailored to ensure the desired properties, the ordinary voter may be left wondering if her vote has actually been counted in the final tally. While auditors may be expected to have access to trusted devices and to understand the concepts behind zero-knowledge proofs, we do not consider these assumptions satisfactory when individual verifiability is considered.

Let us also note the following limiting aspect of the cut-and-choose mechanism in [3,10] and [1]. In all these systems, the audited ballot is handled as a real ballot, but only up to the point when the voter decides to audit it, after which it is discarded. Only one ballot gets to be cast and submitted to the bulletin board. In our proposal, the audited ballot, that we call a *trial* ballot, will be handled in the same way as a real ballot at each phase of the protocol, while additionally playing its role as an audit ballot. In particular, it will be posted on the bulletin board before mixing and handled subsequently in the mix and in the decryption. The traditional cut-and-choose guarantees are recovered in this setting by tracking the trial ballot and the corresponding decrypted vote on the bulletin board before the mix.

4 Trivitas: Trial Credentials and Universal Decryption

The first proposal of Trivitas is the notion of trial credentials. A trial credential is a credential that allows a voter to cast a vote that will not be counted in the final tally, but will appear on the bulletin board at several stages of the tabulation phase. Its purpose is to allow the voter to gain confidence in the correct operation of the system. Trial ballots are identifiable as such only by a threshold set of trustees, thus any component of the system has to treat the set of all ballots in the same way. We show how trial credentials can be implemented in the context of JCJ/Civitas and show their immediate benefit for individual verifiability.

Moreover, we propose another addition to JCJ/Civitas, independent of trial credentials, that brings a further improvement to individual verifiability: we decrypt and publish the content of all ballots after the mix. Therefore, for every ballot that a voter has cast (real, trial or fake), she can verify that the corresponding credential and vote occur on the bulletin board after the mix. This gives a direct evidence to the voter that her ballots, and most importantly her real ballot, have been correctly constructed and processed by the mix network.

4.1 Overview of Proposed Additions

The additions that we propose in each phase are the following. They are also sketched in dotted lines in figures 1 and 2.

Registration. Mirroring the set of real credentials c_1, \dots, c_n we assume that registrars generate a set of *trial credentials* c_1^t, \dots, c_n^t . The set c_1^t, \dots, c_n^t is constructed and distributed to voters following the same protocols as for c_1, \dots, c_n , thus we can assume the same security properties: in particular, trial credentials are indistinguishable from real credentials and fake credentials, for anyone but for the voter that receives its shares. In addition to the electoral roll ER, now we have a *trial roll* TR, that contains the public parts of the trial credentials $\{c_1^t\}_{\text{KT}}, \dots, \{c_n^t\}_{\text{KT}}$.

Voting. In addition to \mathcal{B}_i as in JCJ/Civitas, \mathcal{V}_i constructs a trial ballot $\mathcal{B}_i^t = (\{c_i^t\}_{\text{KT}}, \{v_i^t\}_{\text{KT}}, \mathbf{P}_{\text{corr}}^t, \mathbf{P}_{\text{cv}}^t)$ and uploads both \mathcal{B}_i and \mathcal{B}_i^t to the bulletin board (at implementation level, it has to be decided if a ballot construction form would be used twice or if the system would allow the construction of both ballots at the same time).

Tabulation. At the time of ballot validation (i.e. just after the voting phase ends), the trustees \mathcal{T} additionally perform a PET test of each recorded ballot against the trial roll. For all ballots for which this PET returns true, the trustees decrypt the corresponding credential and the corresponding vote and publish them on the bulletin board: this is the set TrialsA in figure 2. Formally, trustees publish on the bulletin board the result of $\text{PETDecryptUnlink}(\text{CastBallots}, \text{TR})$, where the motivation, specification and the algorithm for PETDecryptUnlink are discussed in section 4.3.

The set of all the cast ballots (that includes the trial ballots) are sent to the mixnet \mathcal{M} for anonymization. Just after the mix and before the PET tests for eligibility enforcement, we propose for all ballots to be decrypted and their corresponding decrypted credentials and decrypted votes to be posted on the bulletin board: this is the set DecBallots in figure 2. To preserve coercion-resistance of the system, this has to be done in a way that does not link the published credentials and votes to the corresponding ballot. We propose the use of a decryption mix $\text{DecryptUnlink}(\text{MixedBallots})$, whose idea is discussed in section 4.3.

At the time of eligibility enforcement, credentials in ballots are additionally tested against the trial roll TR. If a ballot is identified as trial, it is not discarded but is labeled as such on the bulletin board. Finally, all ballots that remain on

the bulletin board after eligibility enforcement are decrypted and only votes that do not correspond to trial ballots are tallied. If a ballot is labeled as a trial ballot, the corresponding credential is also decrypted. The decrypted trial ballots after the tabulation form the set `TrialsB` in figure 2.

4.2 Individual Verifiability in Trivitas

A voter can trace his trial vote in each phase of the system: it should be present on the bulletin board in the set `TrialsA`, after the voting phase, and in the set `TrialsB`, after the tabulation phase. Moreover, relying on the decryption of all the ballots after the mix, the result of which is the set `DecBallots` on the bulletin board, the voter can check that the encryption and the mix has been correct for all of his cast ballots: the one with a real credential, the one with a trial credential and possibly the ones with fake credentials. Hence, fake credentials can also be used for the purpose of end-to-end individual verifiability. In summary, the voter \mathcal{V}_i can check that:

	Verifiability test	Assured property
\mathcal{TV}_1	The pair (c_i^t, v_i^t) occurs in the set <code>TrialsA</code> on the bulletin board	The machine has correctly encoded \mathcal{V}_i 's votes and \mathcal{V}_i 's ballots have been correctly recorded on the bulletin board
\mathcal{TV}_2	The pairs $(c_i, v_i), (c_i^t, v_i^t)$ and all (c_i^f, v_i^f) occur in the set <code>DecBallots</code> on the bulletin board	All of \mathcal{V}_i 's submitted ballots have been input in the mixnet \mathcal{M} and have been correctly processed and output by \mathcal{M}
\mathcal{TV}_3	The pair (c_i^t, v_i^t) occurs in the set <code>TrialsB</code> on the bulletin board	\mathcal{V}_i 's intended vote occurs in the final outcome

Let us argue why all these tests are valid, in the sense that, if they are satisfied for the voter \mathcal{V}_i , then the claimed properties hold with high probability for all of \mathcal{V}_i 's ballots: trial, real and fake. We leave rigorous proofs along the lines of [18,19] as future work, and perform only an informal analysis in the following. As in JCY/Civitas, we assume that either one member of \mathcal{T} is honest or else that auditors check decryption proofs (*). However, this is transparent for the voter, who performs her own verification.

\mathcal{TV}_1 . Assumption (*) ensures that the decryption of trials is correct: the published trial pair is indeed the content of \mathcal{V}_i 's trial ballot, that is present on the bulletin board. Then, the fact that a trial credential is indistinguishable from a real credential ensures that a cheating voting machine or a cheating bulletin board has to make a random guess, thus having at least a 50% probability of being detected.

\mathcal{TV}_2 . Assumption (*) ensures that the set of published pairs (`DecBallots`) corresponds to the decryption of ballots output by \mathcal{M} (`MixedBallots`). Therefore, \mathcal{TV}_2 assures that all of \mathcal{V}_i 's ballots are correctly output by the mix. Moreover, note that \mathcal{TV}_2 increases the assurance offered by \mathcal{TV}_1 , because \mathcal{V}_i can check the correct construction and transmission of all her ballots. Still, \mathcal{TV}_1 is useful to

detect a potential problem as early as possible and also to identify more precisely the elements of the system that have caused the problem. For instance, we will see in the next section how \mathcal{TV}_1 allows for recoverability when a problem is detected before the mix.

\mathcal{TV}_3 . The parallel decryption of trial votes gives some evidence for the voter that votes are not arbitrarily eliminated during eligibility enforcement. This is formally ensured by assumption (*).

4.3 Anonymous PETs and Distributed Decryption with Ciphertext-Plaintext Unlinkability

We now come back to two cryptographic components of the proposed system that have been left out in section 4.1: PETDecryptUnlink and DecryptUnlink. PETDecryptUnlink is used to decrypt trial ballots while keeping them indistinguishable from other ballots. This is necessary for being able to rely on trial ballots to audit the system even after they are decrypted. DecryptUnlink is used to decrypt all ballots, without revealing the link between individual ballots and their content. This is necessary to preserve coercion-resistance: otherwise, a coercer could detect that a ballot cast with a fake credential has been eliminated before the final tally.

Recall that votes and credentials are encrypted with a public key KT , whose corresponding private part $\text{priv}(\text{KT})$ is distributed among \mathcal{T} . In the following, we assume $\mathcal{T} = \{T_1, \dots, T_n\}$. The specification for PETDecryptUnlink and DecryptUnlink is as follows:

PETDecryptUnlink

Input: $\mathcal{S} = (\text{ec}_1, \text{ev}_1), \dots, (\text{ec}_m, \text{ev}_m)$ and $\text{TR} = \text{ec}'_1, \dots, \text{ec}'_k$

Output: $\mathcal{O} = \{(c, v) \mid \exists i_S \in \{1, \dots, m\}, \exists i_{\text{TR}} \in \{1, \dots, k\},$
 $\text{dec}(\text{ec}_{i_S}) = \text{dec}(\text{ec}'_{i_{\text{TR}}}) = c \ \& \ \text{dec}(\text{ev}_{i_S}) = v\}$

Unlinkability: for all $(c, v) \in \mathcal{O}$, the index i_S of $(\{c\}_{\text{KT}}, \{v\}_{\text{KT}})$ in \mathcal{S} is indistinguishable from a random number in $\{1, \dots, m\}$.

DecryptUnlink

Input: $\mathcal{S} = (\text{ec}_1, \text{ev}_1), \dots, (\text{ec}_m, \text{ev}_m)$

Output: $\mathcal{O} = \{(c, v) \mid \exists i_S \in \{1, \dots, m\}. \text{dec}(\text{ec}_{i_S}) = c \ \& \ \text{dec}(\text{ev}_{i_S}) = v\}$

Unlinkability: for all $(c, v) \in \mathcal{O}$, the index i_S of $(\{c\}_{\text{KT}}, \{v\}_{\text{KT}})$ in \mathcal{S} is indistinguishable from a random number in $\{1, \dots, m\}$.

Our proposed implementation for PETDecryptUnlink and DecryptUnlink is an adaptation of the decryption mix idea present in [9][10] to the case of distributed El-Gamal. This setting has already been studied in e.g. [13][12], that show moreover how the shuffle can be made verifiable. However, since we do not require a verifiable shuffle for our application (a misbehavior during decryption would be detected by the voter by simply observing the trial credentials) our algorithms are more straightforward and do not provide zero-knowledge proofs. We only describe the algorithm for PETDecryptUnlink, the second algorithm being similar and more simple.

PETDecryptUnlink. For all et in TR, the parties T_1, \dots, T_n (holding private key shares x_1, \dots, x_n) run the following protocol:

Initial Phase (can be run publicly by any party)

Assume $et = (a, b)$ and, for all $1 \leq i \leq m$, assume $ec_i = (a_i, b_i)$. Compute and publish $p_1 = (\frac{a_1}{a}, \frac{b_1}{b}), \dots, p_m = (\frac{a_m}{a}, \frac{b_m}{b})$. The input for T_1 in the next phase is $(p_1, ec_1, ev_1), \dots, (p_m, ec_m, ev_m)$.

PET Phase (being run privately and consequently by each of T_1, \dots, T_n).

Let $(p_1, ec_1, ev_1), \dots, (p_m^p, ec_m, ev_m)$ be the input for T_i . Create new random numbers $r_1^p, \dots, r_m^p \in \mathbb{Z}_q^*$, $r_1^c, \dots, r_m^c \in \mathbb{Z}_q^*$, $r_1^v, \dots, r_m^v \in \mathbb{Z}_q^*$ and compute

- $(c_1, d_1) = \text{renc}(p_1, r_1^p), \dots, (c_m, d_m) = \text{renc}(p_m, r_m^p)$
- $ec'_1 = \text{renc}(ec_1, r_1^c), \dots, ec'_m = \text{renc}(ec_m, r_m^c)$
- $ev'_1 = \text{renc}(ev_1, r_1^v), \dots, ev'_m = \text{renc}(ev_m, r_m^v)$

Partially decrypt $(c_1, d_1), \dots, (c_m, d_m)$, i.e. compute $d'_1 = \frac{d_1}{c_1}, \dots, d'_m = \frac{d_m}{c_m}$.

Choose a permutation σ of $\{1, \dots, m\}$ and publish

$$((c_{\sigma(1)}, d'_{\sigma(1)}), ec'_{\sigma(1)}, ev'_{\sigma(1)}), \dots, ((c_{\sigma(m)}, d'_{\sigma(m)}), ec'_{\sigma(m)}, ev'_{\sigma(m)})$$

This is the input for T_{i+1} .

Decryption phase (run jointly by T_1, \dots, T_n). For all (p, ec, ev) in the output of T_n : if $p = 1$, perform a distributed decryption of ec and of ev and make the result part of the output set: $\mathcal{O} := \mathcal{O} \cup \{(\text{dec}(ec), \text{dec}(ev))\}$.

If at least one of T_1, \dots, T_n behaves honestly, PETDecryptUnlink satisfies also the unlinkability requirement, as formalized and proved in [12].

5 Other Properties

In this section we discuss other possible applications of trial credentials.

5.1 Universal Verifiability

We propose the following universal verifiability test for Trivitas:

	Verifiability test	Assured property
\mathcal{UV}	All the trials published before the mix are in the set of decrypted ballots after the mix, i.e. $\text{TrialsA} \subseteq \text{DecBallots}$, and they have the same number of occurrences	The mixnet \mathcal{M} is correctly processing all the ballots

We propose this test as an addition to the current universal verifiability proofs, not as a replacement: it is more efficient, but probably offers less assurance than traditional zero-knowledge proofs. On the other hand, this test could also be combined with other tests that offer as well lesser guarantees of correctness but better performance [6], in order to improve their assurance while preserving their efficiency.

Let us argue about the validity of \mathcal{UV} . Because $\text{PETDecryptUnlink}(\text{CastBallots}, \text{TR})$ does not give away what ballots among CastBallots are trials, \mathcal{M} has to treat all the ballots in CastBallots uniformly. In particular, if it chooses to cheat on a subset of ballots in CastBallots , this subset is random. Therefore, if there are enough trial ballots (this could be ensured for instance by letting observers insert any number of trials), a dishonest behaviour of \mathcal{M} would be detected with high probability by the test \mathcal{UV} .

These arguments hold only when the voting machines are not corrupted. Otherwise, a possibly corrupted \mathcal{M} could differentiate trial ballots from other ballots when they are decrypted. We address this problem by a variation of Trivitas that does not let the machine learn which ballots are trials, even when they are decrypted (section 5.3).

5.2 Recoverability from Failed Verification

What happens when individual verifiability fails, e.g. an incorrect trial vote is published along his trial credential? In general, this issue is quite complex, because it requires procedures to determine who is telling the truth: the voter or the voting system. For Trivitas, our proposed recoverability technique is straightforward and requires only a slight modification to the system: trials are decrypted and published in short time after the ballots are cast and the voter does not have to wait for the end of the voting phase to verify a trial. Then, if a voter observes a problem with her trial vote on the bulletin board, she should simply re-vote, using a potentially safer machine. The policy for handling duplicate votes would then be to consider only the last vote as being valid, because it is the vote in which the voter has the highest confidence.

However, like in the case of universal verifiability, this solution is not ideal, because a compromised machine could make a distinction between trial credentials and valid credentials, after trial ballots are decrypted. The variant of Trivitas in the next section addresses this issue.

5.3 The Case of a Compromised Voting Machine

We propose a variant of Trivitas whose aim is to allow universal verifiability and recoverability as discussed above, even in presence of compromised voting machines. The main property of this variant is that it preserves the secrecy of the trial credential, while still allowing the voter to verify a trial vote relying on that credential. The cost is a slightly more complicated voting and vote verification experience:

- along with c and c_t , the voter additionally receives (or constructs) two numbers: one corresponding to a random r and one to $\{r\}_{\text{KT}}$. We may assume that the same protocol is run for obtaining c, c_t and r and hence that the value of r is secret and known only to the voter.
- when constructing a ballot, the voter inputs not only the credential and the vote, but also $\{r\}_{\text{KT}}$.

- when decrypting trial ballots, the trustees \mathcal{T} do not decrypt directly the credential $\{c_t\}_{\mathcal{KT}}$ but instead multiply it with $\{r\}_{\mathcal{KT}}$, to obtain $\{c_t \cdot r\}_{\mathcal{KT}}$ (relying on the homomorphic properties of El-Gamal) and decrypt it to $c_t \cdot r$. Hence, instead of looking for a pair (c_t, v_t) on the bulletin board (like in the basic version of Trivitas), the voter would look for $(c_t \cdot r, v_t)$ (for usability, one can see that an additive homomorphism, also possible with El-Gamal, would be better here).

In this variation of Trivitas, even if the voting machine is compromised, it can not be used to identify which ballots are trials. Hence, trial ballots can also be used for universal verifiability. For recoverability, a trial credential could be used multiple times and the machine would still be forced to take a 50% chance of getting caught each time when it is cheating.

6 Coercion-Resistance

In this section we discuss why Trivitas offers the same coercion-resistance guarantees as JCJ/Civitas. Coercion-resistance in JCJ/Civitas relies on the ability of the voter to create a fake credential c_f and a fake proof P_f that satisfy the following properties:

- given a pair (c', P') , a coercer can not determine if the pair represents a voter's real credential and proof (c, P_f) or if it represents a fake pair (c_f, P_f) . This is due to the fact that at least one registrar is assumed to be honest and the communication channel used with that registrar is assumed to be untappable.
- if a ballot with a fake credential is submitted, it will be eliminated from the final tally in the tabulation phase, during eligibility enforcement. Crucially, all ballots have been mixed and re-encrypted and at least one member of the mix network is assumed to be honest. This ensures that a coercer can not observe to what credentials correspond the ballots that have not been included in the final tally.

As usual, we also have to assume that there are enough votes for each candidate, so that the coercer can not observe that the voter did not follow his instructions from the mere outcome of the election.

The first addition of Trivitas, trial credentials, does not affect the way in which real ballots and fake ballots are handled by the election system. The only observable difference for the coercer is the presence of decrypted trial ballots at every phase and this does not give any information about real ballots or fake ballots. In particular, the two properties mentioned above remain true in presence of trial ballots.

The second addition of Trivitas, universal decryption, is potentially more problematic for coercion-resistance, since it concerns all the recorded ballots. However, coercion-resistance is preserved by two crucial points:

- all ballots are decrypted, without making a difference between real credentials, fake credentials and trial credentials. This ensures that, in Trivitas as in JCJ/Civitas, the coercer can not determine if a credential is valid or not.
- the algorithm applied to decrypt all ballots is a decryption mix, i.e. we apply `DecryptUnlink(MixedBallots)`. It may be surprising that a set of anonymized ballots is decrypted with a decryption mix. However, this is needed because trustees must eliminate fake ballots in a publicly verifiable way. In that case, if the coercer could additionally see the contents of all ballots, he could determine what credentials were invalid.

Towards a Formal Proof. Let us sketch how coercion-resistance proof for JCJ/Civitas [18] could be extended to cover Trivitas. To define coercion-resistance for an election system E in a computational model, [18] considers an ideal system $E_{\mathcal{Z}}$ where the outcome of the election is “magically” computed: the adversary can observe only the final outcome and is not able to influence anything more than vote choices for the compromised voters. Then, a system is said to satisfy coercion-resistance if the probability of a polynomial time adversary being able to determine if it has been cheated is roughly the same when the election is run by E as in the case when the election is run by $E_{\mathcal{Z}}$.

The proof of coercion-resistance is a reduction to (a variant of) the Decision-Diffie Hellman (DDH) assumption: no polynomial time algorithm can distinguish between a Diffie-Hellman tuple (g_1, g_1^x, g_2, g_2^x) and a random tuple (a, b, c, d) . Then, the proof relies on a simulator \mathcal{S} that behaves either as E or as $E_{\mathcal{Z}}$, depending on whether its input is a Diffie-Hellman tuple or not. If there would be a polynomial time adversary that breaks coercion-resistance, i.e. it has better chances of coercion when E is used instead of $E_{\mathcal{Z}}$, then that adversary could be used by the simulator \mathcal{S} to determine if its input is a Diffie-Hellman tuple in polynomial time. This would break the DDH assumption.

To extend this proof to Trivitas all we have to do is to show that the simulator \mathcal{S} of [18] can also execute the additional operations, i.e. the management of trial credentials and the universal decryption of all ballots after the mix. This is possible because the simulator of [18] holds the private key $\text{priv}(\mathcal{T})$ and can therefore decrypt ballots at any time. This makes it possible to simulate both the audit of trial ballots and the universal decryption. Moreover, the same simulator can easily create trial credentials and trial ballots, this process being similar to the creation of real credentials and ballots.

7 Conclusion and Future Work

We have proposed several additions to JCJ/Civitas that improve its individual verifiability. We introduce trial credentials that offer to voters the ability to audit the election process at any stage: creation of ballots, their transmission to the bulletin board, their processing by the mixnet and their final decryption. Moreover, we observe that we can rely on the presence of fake ballots and trial ballots on the bulletin board after the mix to decrypt all ballots without compromising coercion-resistance. This certainly improves individual verifiability: to

our knowledge, this is the first mixnet based system where a voter can directly verify that her actual vote is correctly recorded in the system after the mix. Not only that, but she can also cast as many votes as she likes with fake credentials and check that they are all correctly output after the mix.

The idea of trial ballots is not necessarily specific to JCJ/Civitas. We believe it could be implemented in other electronic voting systems as well, although this requires further research. The universal decryption of ballots after the mix relies on the notion of credentials, to allow voters to identify their votes, and of fake credentials, to allow coercion-resistance. Credentials are also interesting for eligibility verifiability, possible in JCJ/Civitas but generally not possible in other systems. Hence, it would be interesting to investigate the possibility of adding a credential infrastructure on top of other E-voting protocols.

We also plan to develop ideas and variations sketched in section 5. In particular, putting the mechanism for recoverability in the hands of the voter, instead of third party organizations, looks more appealing both from the perspective of the voter and from the perspective of election authorities. Finally, the sketch of coercion-resistance proof from section 6 has to be completed, and this may open other research directions, relating iterative protocol development and the corresponding security proofs.

Acknowledgments. Thanks to Jeremy Clark, Aleksander Essex and anonymous referees for useful comments and interaction on the ideas of the paper. We gratefully acknowledge financial support from EPSRC, through the projects EP/G02684X/1 “Trustworthy Voting Systems” and EP/H005501/1 “Analysing Security and Privacy Properties”.

References

1. Adida, B.: Helios: Web-based open-audit voting. In: van Oorschot, P.C. (ed.) *USENIX Security Symposium*, pp. 335–348. USENIX Association (2008)
2. Adida, B., Neff, A.C.: Ballot casting assurance. In: *USENIX/ACCURATE Electronic Voting Technology Workshop*, Vancouver, BC, Canada (2006)
3. Benaloh, J.: Simple verifiable elections. In: *Proceedings of the USENIX/ACCURATE Electronic Voting Technology Workshop 2006 on Electronic Voting Technology Workshop*, p. 5. USENIX Association, Berkeley (2006)
4. Benaloh, J.: Ballot casting assurance via voter-initiated poll station auditing. In: *Proceedings of the USENIX Workshop on Accurate Electronic Voting Technology*, p. 14. USENIX Association, Berkeley (2007)
5. Blaze, M., Cordero, A., Engle, S., Karlof, C., Sastry, N., Sherr, M., Stegers, T., Yee, K.-P.: Source code review of the Sequoia voting system. In: *Report Commissioned as Part of the California Secretary of State’s Top-To-Bottom Review of California Voting Systems* (July 20, 2007)
6. Boneh, D., Golle, P.: Almost entirely correct mixing with applications to voting. In: Atluri, V. (ed.) *ACM Conference on Computer and Communications Security*, pp. 68–77. ACM (2002)
7. Brandt, F.: Efficient Cryptographic Protocol Design Based on Distributed El Gamal Encryption. In: Won, D.H., Kim, S. (eds.) *ICISC 2005*. LNCS, vol. 3935, pp. 32–47. Springer, Heidelberg (2006)

8. Calandrino, J.A., Feldman, A.J., Alex Halderman, J., Wagner, D., Yu, H., Zeller, W.P.: Source code review of the Diebold voting system. In: Report commissioned as part of the California Secretary of State's Top-To-Bottom Review of California Voting Systems (July 20, 2007)
9. Chaum, D.: Secret-ballot receipts: True voter-verifiable elections. *IEEE Security & Privacy* 2(1), 38–47 (2004)
10. Chaum, D., Ryan, P.Y.A., Schneider, S.: A Practical Voter-Verifiable Election Scheme. In: De Capitani di Vimercati, S., Syverson, P.F., Gollmann, D. (eds.) *ESORICS 2005*. LNCS, vol. 3679, pp. 118–139. Springer, Heidelberg (2005)
11. Clarkson, M.R., Chong, S., Myers, A.C.: Civitas: Toward a secure voting system. In: *IEEE Symposium on Security and Privacy*, pp. 354–368. IEEE Computer Society (2008)
12. 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)
13. 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)
14. 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)
15. 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)
16. Jakobsson, M., Juels, A.: Mix and Match: Secure Function Evaluation via Cipher-texts. In: Okamoto, T. (ed.) *ASIACRYPT 2000*. LNCS, vol. 1976, pp. 162–177. Springer, Heidelberg (2000)
17. Jakobsson, M., Juels, A., Rivest, R.L.: Making mix nets robust for electronic voting by randomized partial checking. In: Boneh, D. (ed.) *USENIX Security Symposium*, pp. 339–353. USENIX (2002)
18. 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 (2005)
19. Kremer, S., Ryan, M., Smyth, B.: Election Verifiability in Electronic Voting Protocols. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) *ESORICS 2010*. LNCS, vol. 6345, pp. 389–404. Springer, Heidelberg (2010)
20. 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)
21. Andrew Neff, C.: Practical high certainty intent verification for encrypted votes (2004)
22. 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)
23. Schneider, S., Llewellyn, M., Culnane, C., Heather, J., Srinivasan, S., Xia, Z.: Focus group views on Pret a Voter 1.0. In: *REVOTE, International Workshop on Requirements Engineering for Electronic Voting Systems* (2011)

The Application of I-Voting for Estonian Parliamentary Elections of 2011

Sven Heiberg¹, Peeter Laud^{1,2}, and Jan Willemson^{1,3}

¹ Cybernetica, Ülikooli 2, Tartu, Estonia

² Institute of Computer Science, University of Tartu, Liivi 2, Tartu, Estonia

³ Software Technology and Applications Competence Center,
Ülikooli 2, Tartu, Estonia
{sven,peeter,janwil}@cyber.ee

Abstract. Estonia has implemented internet voting as a method to participate in various types of elections since 2005. In Riigikogu (parliament) Elections of 2011, over 140,000 voters used the internet voting method. The share of votes cast over the internet among all votes was 24.3%. In light of this popularity it is questioned by various stakeholders whether internet voting can be implemented correctly and securely to support electoral principles such as uniformity. This paper gives an overview of the Estonian Internet Voting System and analyzes events that occurred during the Riigikogu Elections of 2011.

1 Introduction

There are four main voting methods implemented by the Estonian National Electoral Committee (NEC): voting on election day, voting abroad, advance voting and internet voting (i-voting). Most of those methods are paper-based, subject to certain organizational and procedural regulations. Internet voting is exceptional as it offers a possibility for a voter to participate in election digitally from his own computer over the internet anytime during the period of advance voting.

As of August 2011, i-voting has been used in five Estonian elections starting from the Local Government Council Elections in October 2005 and currently ending with the Riigikogu Elections in March 2011. Before the first use, a legally non-binding pilot was conducted in January 2005 in Tallinn. In October 2005, there were 9,317 voters who i-voted (i-voters) and 9,287 counted ballots cast over the internet (i-votes). 1.9% of participating voters and 7.2% of advance voters were i-voters. Since then, the number of i-voters has increased steadily. In March 2011, there were 140,846 i-voters and 140,764 counted i-votes. 24.3% of participating voters and 56.4% of advance voters were i-voters. Those numbers indicate that i-voting is accepted by the electorate. It is also evident that the tally of i-voting has the potential to significantly influence the election outcome.

During the Riigikogu Elections of 2011, several i-voting related incidents took place. Appeals were filed to NEC demanding that the i-voting results should be

revoked because of alleged vulnerabilities of the applied scheme and the legislative problems [2–4]. In one case, the revocation of election results altogether was demanded [14]. All those appeals were dismissed by the Supreme Court [20–22].

The history of Estonian i-voting goes back to 2001, when two reports concerning i-voting were published. The analysis ordered by the Ministry of Justice [16] stated that it was unrealistic to implement statewide i-voting in 2002 because of the lack of suitable technology; instead, a research program towards i-voting was suggested. The analysis ordered by the Ministry of Transport and Communications [23] suggested that it was possible to implement statewide i-voting in 2002. Both reports agreed that i-voting is inevitable in the future and development thereof should be considered as a long term process.

In 2002, i-voting was regulated in the Riigikogu Election Act with the condition that the method shall not be applied before 2005. In 2003, a group of experts proposed an i-voting scheme for Estonia [7]. The security analysis of the scheme stated that in order to implement i-voting, it was necessary to find optimum between the theoretical security of the voting scheme and the complexity of its implementation. On one side of the compromise, there were comprehensibility, similarity to conventional voting, maximal use of the digital signature solutions available in Estonia, simplicity of the cryptographic protocol, and feasibility to implement the system with the know-how present in Estonia itself [6].

The analysis also stated: “The other side of the compromise or, in principle, the weak point of the scheme, is the need to trust central servers and computers of the voters. Is such a compromise reasonable? In our opinion – yes.” [6]

A few months later, in 2004, another group of experts published a report which analysed computer and communication security issues in the internet-based voting system SERVE (Secure Electronic Registration and Voting Experiment) built for the U.S. Department of Defense. The report stated that internet- and computer-based voting systems have numerous fundamental security problems, which open the possibility for large scale attacks such as selective voter disenfranchisement, privacy violation, vote buying and selling and vote switching [13].

The authors of the SERVE-report made a new public statement in 2007 where they found another Department of Defense report on electronic voting technology quite troubling and assured that the arguments presented in 2003 still hold and there is no way to secure internet voting [12].

In 2010, a modified version of security analysis for the Estonian i-voting scheme found that the challenge of successful i-voting has been solved and implemented in practice [5].

OSCE/ODIHR Election Assessment Mission observed the Estonian 2011 parliamentary election with i-voting as one focus. The final report of the mission gave altogether 13 recommendations regarding to the legal framework, oversight and accountability, and some technical aspects of the internet voting system [19].

In subsequent sections, we will give an overview of the i-voting scheme implemented in Estonia, explain its architecture and approach to security, give an

overview of the most important incidents during Riigikogu Elections of 2011, and propose a road ahead based on the analysis of these incidents.

2 Estonian I-Voting Scheme

2.1 Legal and Technical Framework

Requirements for the design and implementation of a voting method come from the legislation which defines the election and the electoral system. The Constitution of the Republic of Estonian states the basic electoral principles – freedom, generality, uniformity, directness and voting by secret ballot. The Riigikogu Election Act defines Riigikogu Elections in detail by regulating candidates, registration, voting procedures, etc. Similar legislation exists for other types of elections.

Since 2002, Estonian citizens are issued National Identity Cards (ID-cards) carrying a chip capable of performing RSA operations. Each chip contains two RSA secret keys, used respectively for authentication and for signing. The corresponding public keys are bound to the cardowner by certificates issued by the Estonian National Certification Authority (NCA). Various government and private sector services use the ID-cards for identification and legally binding digital signatures. Currently, the ID-card is the primary identity document for both computer mediated and direct communication which makes it available to all voters. The principle of using ID-cards to do i-voting has been present in Estonian legislation since 2002.

To fight bribery and coercion, the concept of i-vote revocation is legislated. A voter can cast an i-vote several times, only the last one will be counted. I-vote is also revoked if the voter uses any paper-based voting method (casts a p-vote) during the advance voting period. Those measures were not accepted unanimously by the Riigikogu; there were doubts whether it violates the uniformity of elections and secrecy of ballots [18]. The objection was that paper-based voting methods do not allow revocation. As a mild compromise, the law limited the possibility for i-vote revocation to the advance voting period.

The Penal Code has defined several election-related criminal offences such as interference with election. NEC has authority to declare the voting invalid on polling division, electoral district, county or state level if some detected violation of the law significantly affected or may significantly affect the voting results. In this case, repeat voting is held. I-voting results can currently be declared invalid only as a whole. If the i-voting is cancelled due to some violation before the actual election day, then the electorate is notified and voters can revoke. In this case, no repeat voting is held.

2.2 Architecture of the I-Voting Scheme

We shall describe the architecture of Estonian i-voting scheme only as much as it is necessary to understand the case studies in the subsequent sections. Readers can refer to [7] for a more complete description.

The core components of the architecture of the i-voting scheme are i-voting protocol, i-voting system and i-voting client application (IVCA) - an election specific application which allows voters to cast their votes from Windows, Linux and MacOS X based computers.

I-voting system (IVS) is responsible for i-vote collection, storage and tabulation. The system is interfaced with NCA, Election Management System, Population Register and NEC website. The roles of the IVS are fulfilled by three different servers:

- The Vote Forwarding Server (VFS) is responsible for authenticating i-voters, distributing candidates' lists to i-voters and accepting the i-votes; VFS is available over the internet.
- The Vote Storing Server (VSS) is responsible for storing the i-votes over the period of time and for the anonymization of the i-votes before the actual tabulation; VSS is kept behind a firewall, connections from VFS are allowed.
- The Vote Counting Server (VCS) is responsible for the tabulation process. VCS is offline at all times.

IVS has a RSA keypair to protect ballot secrecy. The public key (ivs_{pub}) is published with IVCA. The private key is stored in tamper-resistant hardware security module (HSM) used only by VCS and protected by multiparty authentication scheme. In practice, 4 of 7 NEC members have to be present to activate the private key.

For Riigikogu Elections, each voter belongs to one of 12 electoral districts. Each candidate has a unique candidate number and is registered to one electoral district. Only voters from the same district can vote for the candidate. We refer to the list of candidates that voter v can vote for as C_v .

Setup Phase. I-voting takes place in four phases: setup, voting, revocation, and tabulation. In the setup phase, the IVS is prepared for election. The VSS is set up with the list of voters and an empty digital ballot box. The VCS is set up with the list of candidates. The IVS keypair is generated in HSM. The VFS is set up with the list of voters, the list of candidates, and HTTPS authentication certificate. The IVCA is set up with election specific data including ivs_{pub} and VFS certificate. The IVCA is digitally signed by NEC; fingerprints and download location are published in newspapers and on the NEC website.

Voting Phase. In the voting phase, the i-voting protocol is executed between the IVCA and the VFS. Mutually authenticated HTTPS is used as the transport protocol. The VFS verifies that v is an eligible voter and returns the candidate list C_v with an indication whether it is a repeated vote or not. After the voter v has selected a candidate $c \in C_v$, encryption is used to produce an anonymous ballot: $b_{anon} = RSA_{enc}(ivs_{pub}, c)$. The anonymous ballot is signed with the voter's ID-card. The i-vote consisting of b_{anon} , a digital signature and a signing certificate of v is sent to the VFS which verifies the signature and forwards the ballot to the VSS for storage. The VSS verifies the signature and checks

the status of the signing certificate by NCA. If no problems occur, the i-vote is stored before revoking any possible previous i-votes cast by the voter. HTTP is used as the transport protocol between the VFS and the VSS.

At the end of the voting phase, the list of all i-voters is generated and sent to the polling stations for reference. The VFS and the VSS are disconnected from the network.

Revocation Phase. The voting phase is followed by the revocation phase during which i-votes of those who also have p-voted are revoked.

Tabulation Phase. At the end of the revocation phase, the contents of the ballot box are anonymized – digital signatures are separated from encrypted votes so that the VCS will not be able to see which voter voted for which candidate. Anonymized ballots are stored in 12 distinct lists according to the election districts that the original voters belonged to. Those lists are burned to a DVD and carried to the VCS. For the tabulation itself, the IVS’s private key is activated and the anonymous ballots are decrypted. After the decryption, valid candidates for the district are tabulated.

2.3 Security Considerations

The Estonian i-voting scheme relies on the assumption that we can trust the owner of the IVS and we can trust voter’s computer. However, some measures to reduce the necessary trust have been taken.

Trust in the IVS. For electoral principles to hold, the IVS has to function correctly: accept all of the votes cast by eligible voters, preserve the integrity of the ballot box at all times, anonymize the votes before the tabulation, correctly execute the correct tabulation algorithm on the correct input and publish the produced output. To achieve this correct behaviour, a set of organizational regulations and procedures are established, all of which are audited. For example, the anonymization of i-votes can only occur in the presence of at least 2 election officials, an auditor and possible external observers. All procedures are defined beforehand in written form, and all actions and outcomes are recorded on tape. Without enforcing those regulations, the IVS owner could manipulate the election results on a large scale by adding or removing votes from the digital ballot box without getting caught.

To support organizational protocols responsible for ballot box integrity, five audit logs containing SHA1-checksums of i-votes calculated over b_{anon} are stored in the VSS and the VCS. The contents of the log-files are following:

- L_1 : checksums of all i-votes accepted by VSS;
- L_2 : checksums of all i-votes revoked by VSS;
- L_3 : checksums of all i-votes sent to tabulation by VSS;
- L_4 : checksums of all i-votes declared invalid by VCS;
- L_5 : checksums of all i-votes declared valid by VCS.

L_1 , L_2 and L_3 also contain the personal code of the voter. This means that a person who has access to all of the audit logs is able to link the original voter to the hashvalue of encrypted ballot b_{anon} . At the end of the tabulation phase, the following conditions must hold:

$$L_1 = L_2 \cup L_3 \text{ and } L_2 = L_4 \cup L_5 . \quad (1)$$

If those conditions do not hold then it can be said that the contents of the digital ballot box have been tampered with.

Trust in Voter's Computer. The Estonian i-voter has to trust the computer used for i-voting with the IVCA. Malicious software executed in the computer could manipulate the IVCA to break secrecy and integrity of the ballot. If the malicious software could be distributed widely, the attacker would have the potential to manipulate election results on a large scale. This could occur in several ways, e.g. by sending modified votes to get his candidate elected or by sending encrypted garbage to discredit the i-voting altogether.

Detection-Based Security. The state of the art in malware distribution leaves no doubt that the environment where the IVCA is executed cannot generally be considered safe. Malware can use several attack-vectors to alter the behaviour of the IVCA with no feedback to the voter. One possible vector would be using debugging interfaces offered by the underlying operating system. Debugging a process means stepping through its instructions one at a time while examining the contents of the memory. It is common that while debugging a process, a developer decides to overwrite some memory locations with new values, thus altering the actual state of the process. A similar approach can be taken by malware attacking the IVCA: if a voter has selected the candidate c_1 then change the selection to c_2 right before the encryption. The success of the attack depends on the capability of finding the right breakpoint to stop the process and finding the right memory location to alter. With the lack of the IVCA source code, an attacker can approach those problems by reverse engineering the executable file of the IVCA.

It is possible for a process to detect whether it is being debugged. An attacker who by reverse engineering discovers that some detection is used can take countermeasures in his malicious code.

To reduce the risk that a vote cast by the IVCA is tampered with by malicious code to an acceptable level, the following actions have been taken: (i) a detection system for known attack-vectors is built into IVCA; (ii) methods are used to complicate reverse engineering. During the voting phase, the IVCA instances report their opinion on the hostility of the environment to the IVS. The NEC, CERT-EE and volunteers from the Estonian Cyber Defence League also monitor the IVS and Estonian internet for known malware activity. This information is input to the NEC to decide whether i-voting is under attack or not.

It is hoped that a 7-day i-voting period is short enough to avoid reverse engineering of the IVCA, designing and implementing robust and stealthy malicious code, distributing and activating it on a large scale.

3 Riigikogu Elections of 2011

In this section, we study two cases which occurred during the Riigikogu Elections of 2011. On February 26th, a student turned to NEC claiming that he had written an election rigging malware which was able to tamper with the IVCA. On March 6th, during the tabulation phase, one of the i-votes was declared invalid. Those two cases are not directly related to one another but indicate possible problems with the current i-voting scheme and therefore deserve some analysis.

3.1 Case: Invalid I-Vote

Invalid I-Vote Is Found. One of the i-votes was registered invalid by the VCS during the tabulation phase of the Riigikogu Elections on March 6th, 2011. In the case of p-voting, an invalid vote is nothing special. It is quite common that voters cast invalid votes intentionally to express the attitude towards the ongoing election by leaving the ballot paper empty or writing different free-text statements on it.

The Estonian electoral system does not give any meaning to invalid votes; they are not considered as part of the election result. The IVCA has no functionality for casting an empty or otherwise invalid ballot. A voter who wants to intentionally cast an invalid i-vote, must write a new IVCA that makes it possible to encrypt random data, or find a way to manipulate the current IVCA to cast an invalid vote.

Analysis of the Cause. The possibility that an i-vote could be invalid was foreseen in the i-voting protocol. Also, the software was developed to distinguish valid i-votes from invalid ones. On the other hand, this was the first time when one of the i-votes was found invalid, so a bug in the software or the procedures for handling the IVS during the election was suspected.

The analysis of the VCS error logs showed that the invalid i-vote appeared to be correctly encrypted with *ivs_{pub}*. This left two conceptual possibilities for the vote to become invalid: (i) the plaintext did not follow the formatting rules for i-vote; (ii) the plaintext followed the formatting rules, but pointed to a non-existent candidate number. Further analysis pointed out five possibilities for the invalid vote to occur:

- a bug in the IVCA – sending a malformed ballot to encryption;
- a bug in the VFS – sending an invalid candidate list to the IVCA;
- a bug in the VCS – misinterpreting the decrypted vote;
- human mistake – the VCS and the VFS were set up with incompatible candidate lists;
- someone intentionally cast an invalid i-vote.

I-vote decryption seemed to be necessary to rule out most possibilities. It would be relatively safe to say that there is a bug in the VCS if the plaintext was a valid vote pointing to a valid candidate and the candidate lists in the VFS and the

VCS were compatible. Only human mistake in the IVS setup procedures could be excluded without decrypting the i-vote. On the grounds of this analysis, the NEC decided on April 1st to decrypt the invalid i-vote and examine its contents. The decision was later reverted due to the possible threat to electoral principles.

Ballot Secrecy. The Constitution of the Republic of Estonia holds voting by secret ballot as one of the main electoral principles. To achieve this requirement, RSA encryption is used. To achieve another principle – uniformity – b_{anon} is stored together with the digital signature which unanimously identifies the original voter. To maintain ballot secrecy, i-votes are anonymized before the tabulation.

The fact that the IVS auditing logfiles can link original voters to the hashes of the encrypted ballots means that if the invalid i-vote was indeed decrypted separately from other i-votes, the ballot secrecy would be protected only by organizational means.

Invalid I-vote as a Possible Attack. In parallel to setting up the analysis framework, additional tests and preliminary code reviews to the IVS were conducted. No bugs were identified and the possibility of an intentionally spoiled vote was taken into consideration. It occurred that there is at least one relatively easy way to influence the contents of b_{anon} without writing a new application or directly attacking the existing one.

If the ID-card is used for i-voting, then the HTTPS connection between the VFS and the IVCA is mutually authenticated. In the case of cell phone based digital identity – Mobile-ID – only the VFS is authenticated; the voter identification follows from the Mobile-ID protocol and cannot be used on the HTTPS level. On the Windows platform the trust to the VFS is configured via the system certificate stores – if the HTTPS certificate of the VFS is signed by some trusted certification authority (CA), then the connection is trusted. This opens the possibility for a man-in-the middle attack where the user’s certificate store is compromised with the attacker’s CA certificate and an intercepting HTTPS proxy using a certificate signed by attacker’s CA is installed between the IVCA and the IVS. The proxy modifies the original candidate list sent to the IVCA so that it contains invalid candidate numbers. This is possible because the candidate list sent from the VFS is not digitally signed as the HTTPS channel security is considered sufficient to guarantee the integrity of the message. The user does not notice the invalidity of the candidate numbers and casts a vote which is correctly formatted and encrypted by the IVCA and forwarded to the VFS by proxy. This type of intentionally invalidated i-vote would have been falsely identified as a bug in the IVS after the i-vote decryption.

If the attacker wasn’t aiming for the discrimination of the voter but for publicity, then the previous scenario would be used by the attacker himself to decoy the election officials to show whether the NEC can find out who did cast the vote from the contents of the ballot. If some more sophisticated technique to invalidate the ballot would have been applied, then the contents of the ballot could have been anything from the personal identification of the attacker or

personal identification of someone not involved at all to a well formed ballot with an invalid candidate number.

Reverting the Decision. After considering the matter of ballot secrecy and the possibility of an attack against i-voting as such, the NEC reached the conclusion that it would be better not to create a precedent of decrypting one i-vote separately from others. The decision from April 1st was reverted on April 8th. It was decided to carry through only those analyses that do not require the ballot decryption. After the analysis, it was clear that there were two possibilities: (i) a hard to find bug (such as memory corruption) in the IVCA, or (ii) an intentionally spoiled i-vote.

3.2 Case: Student Writes a Ballot-Manipulating Script

Revocation of I-Voting Results Demanded. On February 26th, student P. sent an e-mail to the NEC and three major newspapers claiming that he had written a prototype of an election rigging malware. Attached to the e-mail there was a presentation which referred to the SERVE report [13]. The presentation pointed out that a malicious piece of software controlling both input and output interfaces on a client computer was a threat to the IVCA as it was capable of manipulating the voter to believe that he has voted for candidate c_1 , although the malware actually voted for candidate c_2 .

Election officials analyzed logs of both ongoing election and the test-election from February 8th to 10th. It appeared that the most remarkable voting session from the test-election belonged to P. The session lasted for 14 hours and indicated several attempts to tamper with the IVCA. Also, every time P. had voted during the real election, his voting sessions were marked suspicious.

On February 28th, P. gave election officials access to the source code of his malware. P. also demonstrated the attack to journalists. Election officials claimed that the type of malicious behaviour implemented by the malware was detected by the IVS. In response to these claims P. implemented a new type of attack – the malware now selectively held back ballots for certain candidates, whereas the voter was left with the impression that his vote was successfully sent to the VFS. This attack was demonstrated to observers of OSCE/OHDIR on March 1st, and it was screened on National Television on March 9th after the election had ended.

On March 5th, P. filed an appeal to the NEC [3] demanding the revocation of all i-votes, claiming the following:

- Ballot secrecy was not guaranteed as the IVCA can be a subject to screen monitoring software;
- The IVS contained no protection against voter disenfranchisement type of attacks;
- CERT-EE’s capability of monitoring malware distribution in the Estonian internet was not proven;
- The voter cannot check whether his vote was accepted by the IVS, hence the system did not comply with Riigikogu Election Act.

On March 6th, preliminary election results were published. On the same day, the NEC decided to reclassify the appeal as a note as it did not address any violations of personal rights. A reply was sent to P. clarifying that the NEC is aware of potential attack objects, methods and time, which makes the task of detection significantly easier and no attempts to attack the IVS have occurred in any election. The NEC assured its awareness of alternative protocols reducing the need to trust the voter's computer and did not exclude their use in the future. The NEC stated that the IVS has been implemented to be trustworthy and compliant to Riigikogu Election Act [8].

P. modified his original appeal and refiled it to the NEC on March 8th, still demanding the revocation of all i-votes [2]. The NEC, according to its procedures, forwarded the appeal to the Supreme Court which on March 21st dismissed the appeal arguing that although P., as an adult Estonian citizen, could be subject and therefore an interested party to voter disenfranchisement attacks, himself knowingly executed this type of malware in his computer, therefore his rights were not violated. It is necessary to detect the violation of the person's right to vote in order to to revoke the election results, hypothetical possibility alone is not sufficient for the revocation [21].

The actions of P. did not go unnoticed. On March 24th, one of the parliament-parties filed an appeal and demanded the revocation of the election results as a whole [14]. The appeal referred to findings of P. and was dismissed by the Supreme Court on March 31st [20].

Technical Details of the Attack. The initial version of the malware attacked both the ballot secrecy and integrity; the version presented in the television attacked ballot secrecy and implemented voter disenfranchisement based on the candidate selection. P. gave the NEC access to the initial version of the malware, therefore, this version is described here. The modifications from the initial version to the version presented in television should be relatively straightforward to anybody with some programming experience.

In its general setup, the attack was related to the attack of Estehghari and Desmedt against Helios internet voting system [9]. Both of the approaches modify the appearance of the voting software on screen. However, since Helios is a web-based application, Estehghari and Desmedt were able to mangle with the document object model of the webpage to achieve the desired result. The proof of concept malware of P. used the IVCA graphical user interface (GUI) as an attack-vector. It was written in AutoIt scripting language [1] which is a framework for scripting GUI-based Windows applications. Optical character recognition (OCR) technology was used on the IVCA screenshots to decode voter personal data and the intended candidate. Fake-IVCA was built from the screenshots and the message-loop of the original IVCA was poisoned with generated mouse events. The fake-IVCA was used to leave the voter with the impression that the ballot was cast as intended. Underneath the fake-IVCA, the original software accepted generated events and voted for a semi-randomly selected candidate.

The malware took advantage of the fact that the IVCA is a wizard-like application with a very simple state machine. Although it is possible to move back and forth between various views, most people cast their ballot in one go. The malware was implemented as a series of stages, whereas each stage and corresponding actions were associated with a certain state of the IVCA. The success of the malware depended on its ability to detect state changes in the IVCA and hide its own existence from the voter.

To detect state changes, certain IVCA GUI regions were examined with AutoIt PixelChecksum function. If a voter reached the candidate selection, the ballot manipulating code was activated. The fake-IVCA – a screenshot of the candidate list and the selected candidate – was hiddenly generated. Regions of the original IVCA were examined to determine whether the voter had selected another candidate and the fake-IVCA was updated respectively. When the voter pressed a button to encrypt and sign the ballot, the fake-IVCA was made visible on top of the real IVCA. Mouse events were generated to the fake-IVCA to select any other candidate. This was possible as the fake-IVCA consisted of a non-transparent clickthrough foreground window which accepted mouse events but passed them without modification to the window underneath it. This way, the main window of the original IVCA received the events, but the foreground indicated no changes in the application state. After the encrypted and signed ballot had been successfully sent to the VFS, the original data was analyzed with OCR and saved to a log file.

Unlike the attack by Estehghari and Desmedt [9], the malware of P. did not contain a distribution mechanism and it did not hide its existence nor behaviour any more than it was necessary for a prototype. For example, the GUI checksum method was not robust enough to be applied over a large set of computers with different screen resolutions, operating system versions, display drivers and installed fonts; fake-IVCA creation and OCR methods used the file system to store the data, no steps were taken to hide those files, and the mouse event insertion method to modify the ballot was not robust enough to vote for a specific candidate. Although these shortcomings rendered this specific piece of malware useless for a real large scale attack, it is possible to overcome them with a reasonable amount of extra effort.

4 Discussion

Besides the two major issues discussed in Section 3, there were others as well. Three people turned to the i-voting help-desk with the following problem: the IVCA GUI was too large to fit onto their computer screen and two candidates on the bottom of the list were hidden by the Windows task-bar. The problem was caused by fixing the minimal supported resolution for the GUI design. This event was picked up by one of the candidates who demanded nullification of i-voting results [4].

The public reaction to this shortcoming in the IVCA shows clearly that for i-voting it does not suffice to be clean; it also has to look clean. Any shortcomings in quality of the system and transparency of the processes have a potential to become a weapon in political battles. The abovedescribed shortcoming in the IVCA GUI is easily avoidable in the next version of the IVCA, whereas the two cases presented in Section 3 identify conceptual problems in the Estonian i-voting scheme.

Invalid I-Vote. By now, it is clear that the root cause of the invalid i-vote will never be exactly known. Although several initial versions were excluded in the process, the distinction between a bug in the IVCA or intentionally invalidated i-vote could not be made. Minor technical corrections to the i-voting protocol and the IVS do not change the fact that the ballot secrecy relies heavily on organizational procedures in the Estonian i-voting scheme. It is theoretically possible for the NEC not to anonymize i-votes and use a modified VCS to break the secrecy of all ballots. To break the secrecy of one ballot, it is sufficient to decrypt it separately from others and later analyze audit log-files.

The case of an invalid i-vote could have been avoided with an i-voting protocol using zero-knowledge ballot validity proofs [15]. The validity of i-votes could have been verified, while the identity of the voter was still known, leaving more room for the action to NEC. On the other hand, this raises new questions such as the legality of validating the contents of the ballot before storing it in the digital ballot box. An IVS implementing a mix network-based anonymization system [15] would have reduced the required trust in the NEC and allowed the analysis of invalid i-vote in a secure manner with respect to ballot secrecy.

Student's Attack. P. exploited the fact that the Estonian i-voting scheme contains no hard countermeasures against malicious computer. The anomaly detection system only makes it possible to indicate that something is happening, and it is not possible to exclude the possibility of something happening. Any detection-based protection scheme has the following weaknesses:

- an attack with no signature in the detection engine is undetected;
- the efficiency of the scheme relies on the actual response to the detected incidents.

As P. used a previously known attack-vector, his tampering with ballots was detected by the IVS, but there were no prompt mechanisms to discover the voter disenfranchisement attack. Neither was the IVS actively monitoring the percentage of i-voters not finishing their transactions, nor offering a proof to a voter that the ballot cast was accepted as intended. The lack of countermeasures against voter disenfranchisement allowed P. to execute his attack successfully.

Most sessions associated with P. were marked as suspicious by the detection engine, but no action besides observation was taken by the NEC until the attacker wrote an e-mail himself. Then the communication with the attacker became possible and the NEC got to analyze the reasons for the alerts in the log-files.

Currently, the only real action that the NEC can take in case of large scale attacks against the IVS, is to revoke i-voting results as a whole and call people to p-voting. A single anomaly has no significant impact to the election results and is therefore not acted upon. In light of more than 140,000 i-voters from more than 100 countries this lack of repertoire is dangerous.

Toward Secure I-Voting. As with the IVS server side problems, there exist i-voting protocols which handle the problem of trusting voter's computer.

The IVCA is executed in the malicious environment, where malware could manipulate its behaviour. One possible solution is to use a blind voting scheme such as one proposed by Okamoto [17] where for each voter personalized candidate numbers (codes) are generated. Codes can later be re-unified by the tabulation process for tally. The voter gets his codes through some pre-channel and uses computer to enter and send the code for the desired candidate. It is impossible for a malware to know all the codes which leaves denial of service as the main attack-vector. The problems with this protocol are (i) one cannot i-vote without the codes; (ii) code generation must be performed in a privacy preserving manner; (iii) most of the people will not find this system usable.

Tamper-indicating voting schemes such as [10] and [11] take a weaker approach to security. Each voter again gets personalized candidate numbers (codes) over a pre-channel. Each voter also has a post-channel with the IVS such as SMS. After voting with a point-and-click GUI, the IVS sends a receipt to the voter over the post-channel. The receipt can then be compared to the voter's codes to see if it matches the candidate the voter intended to vote for. In this scheme, a malware can still manipulate the ballot, but will be detected when doing so. This is enough to detect manipulations on a large scale which is crucial in assuring free election.

From the viewpoint of ballot secrecy, tamper-indicating voting schemes run into conflict where the IVS sends the voter a code corresponding to the candidate the voter voted for, but the NEC claims that the voter's identity and the contents of the ballot cannot be connected. Cryptographic protocols achieve this property of 'knowing without knowing' for example with zero-knowledge proofs or homomorphic encryption. If these methods are not understandable to the general public, this property can be used as a weapon in political battles.

The multi-channel nature of tamper-indication needs further analysis to clarify the security requirements and explain the risks if the requirements are not met. For example, if it would be possible to control both the code generation and the post-channel, a large scale attack against ballot secrecy would be possible.

Multi-channel protocols also have a higher computational cost which must be analyzed in light of real life requirements. The protocol of [11] does not scale for the Estonian case of ≈ 800 candidates and $\approx 900\,000$ voters. Gjøsteen [10] notes that protocols based on homomorphic tallies are not efficient for Norwegian elections. Organizational complexity which requires independent parties to host separate components of the system makes it difficult for a single governing body to organize elections.

Tamper indication is another method of detection where the question of a legal framework for incident response is as important as with the current scheme. The possible actions that the NEC can take, in case an anomaly is detected, have to be regulated.

The protocol [11] is also subject to false-positives which introduces a new attack-vector. Consider the voter claiming that he has voted for candidate a and has in fact received the code for candidate b . The SMS contains information for detection, but detection itself is not a proof. If we can only detect the manipulation and not prove it, this opens new ways for manipulation. On the other hand, if a voter is able to prove that his vote was accepted as intended, bribery becomes possible which is dangerous in complex social situations.

Avoiding bribery and at the same time avoiding malware is an example of contradictory requirements that an i-voting system must fulfil. If there is no method to completely satisfy both requirements, a political decision has to be reached about which risk is acceptable in the given context. For these decisions, the i-voting requirements must be considered as a whole. The treatment of one vulnerability in isolation from other requirements will not result in a functional i-voting system.

5 Conclusions

Estonia has put a lot of effort into developing a usable and efficient i-voting system. The i-voter turnout in Riigikogu Elections of 2011 shows that the Estonian electorate has accepted i-voting as a voting method. In this article, we described the Estonian i-voting scheme and discussed how it complies with the electoral principles – a fair amount of the i-voting architecture is concerned with meeting the security requirements deduced from those principles.

We saw that during Riigikogu Elections of 2011, several weaknesses present in Estonian i-voting scheme were materialized. The analyzed events indicate real-life attacks that an i-voting system has to withstand. From these events we conclude that it is necessary to work toward new, more secure i-voting protocol. We need to reduce the level of trust required in the voter's computer and provide the NEC with means to show that it could not act malicious even if it wanted to. It is possible that i-voting related legislation may be refined to meet these requirements.

The goal of secure i-voting cannot be reached by dealing with single vulnerabilities in isolation. Requirements for an i-voting scheme must be handled as a system. Due to the interdisciplinary and possibly contradictory nature of the system, both the creation of the system and the design of the technical solution according to the system require a widely-accepted political decision. It is also important to explain the system and the possible choices among all options to the general public in order to reduce the risk of manipulation with the election results.

References

1. AutoIt Automation and Scripting Language, <http://www.autoitscript.com/site/autoit/>
2. Appeal no. 14-11/406-2 to NEC (in Estonian) (March 8, 2011), <http://www.vvk.ee/valimiste-korraldamine/vabariigi-valimiskomisjon-yld/kirjad>
3. Appeal no. 14-11/406 to NEC (in Estonian) (March 5, 2011), <http://www.vvk.ee/valimiste-korraldamine/vabariigi-valimiskomisjon-yld/kirjad>
4. Appeal no. 14-11/446 to NEC (in Estonian) (March 10, 2011), <http://www.vvk.ee/valimiste-korraldamine/vabariigi-valimiskomisjon-yld/kirjad>
5. Ansper, A., Buldas, A., Jürgenson, A., Oruaas, M., Priisalu, J., Raiend, K., Veldre, A., Willemson, J., Virunurm, K.: E-voting Concept Security: Analysis and Measures. Estonian National Electoral Committee, EH-02-02 (2010)
6. Ansper, A., Buldas, A., Oruaas, M., Priisalu, J., Veldre, A., Willemson, J., Virunurm, K.: E-voting Concept Security: Analysis and Measures. Estonian National Electoral Committee, EH-02-01 (2003)
7. Estonian National Electoral Committee. E-Voting System. General Overview (2010)
8. Estonian National Electoral Committee. Answer to note 14-11/406 (in Estonian) (March 7, 2011), <http://www.vvk.ee/valimiste-korraldamine/vabariigi-valimiskomisjon-yld/kirjad>
9. Estehghari, S., Desmedt, Y.: Exploiting the Client Vulnerabilities in Internet E-voting Systems: Hacking Helios 2.0 as an Example. Helios (Section 4), 0–27 (2010)
10. Gjøsteen, K.: Analysis of an internet voting protocol. Cryptology ePrint Archive, Report 2010/380 (2010), <http://eprint.iacr.org/>
11. Heiberg, S., Lipmaa, H., van Laenen, F.: On E-Vote Integrity in the Case of Malicious Voter Computers. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) ESORICS 2010. LNCS, vol. 6345, pp. 373–388. Springer, Heidelberg (2010)
12. Jefferson, D., Aviel Rubin, D., Simons, B.: A comment on the May 2007 DoD report on Voting Technologies for UOCAVA Citizens (2007), http://www.servesecurityreport.org/SERVE_Jr_v5.3.pdf (last accessed on August 27, 2011)
13. Jefferson, D., Aviel Rubin, D., Simons, B., Wagner, D.: A Security Analysis of the Secure Electronic Registration and Voting Experiment (SERVE) (2004), <http://www.servesecurityreport.org/paper.pdf> (last accessed on August 27, 2011)
14. MTÜ Eesti Keskerakond. Appeal 14-12/535 (in Estonian) (March 24, 2011), <http://www.vvk.ee/valimiste-korraldamine/vabariigi-valimiskomisjon-yld/kirjad>
15. Lipmaa, H.: Secure Electronic Voting Protocols. In: The Handbook of Information Security, vol. II. John Wiley & Sons (2006)
16. Lipmaa, H., Mürk, O.: E-valimiste realiseerimisvõimaluste analüüs (An Analysis of the Possibility to Organize E-voting), Analysis ordered by Estonian Ministry of Justice (2001) (in Estonian)
17. 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)

18. OSCE/ODIHR. Republic of Estonia. Parliamentary Elections (March 4, 2007). OSCE/ODIHR Election Assessment Mission Report, ODIHR. GAL/56/07 (2007)
19. OSCE/ODIHR. Estonia. Parliamentary Elections (March 6, 2011). OSCE/ODIHR Election Assessment Mission Report (2011)
20. Rask, M., Põld, J., Salmann, H.: Decision of Supreme Court 3-4-1-10-11 (in Estonian) (March 31, 2011), <http://www.vvk.ee/valimiste-korraldamine/vabariigi-valimiskomisjon-yld/kirjad>
21. Rask, M., Põld, J., Salmann, H.: Decision of Supreme Court 3-4-1-4-11 (in Estonian) (March 21, 2011), <http://www.vvk.ee/valimiste-korraldamine/vabariigi-valimiskomisjon-yld/kirjad>
22. Rask, M., Põld, J., Salmann, H.: Regulation of Supreme Court 3-4-1-6-11 (in Estonian) (March 23, 2011), <http://www.vvk.ee/valimiste-korraldamine/vabariigi-valimiskomisjon-yld/kirjad>
23. Tammet, T., Krosing, H.: E-valimised Eesti Vabariigis: võimaluste analüüs (E-voting in Estonia: Feasibility Study), Analysis ordered by Estonian Ministry of Transport and Communications (2001) (in Estonian)

Towards Best Practice for E-election Systems

Lessons from Trial and Error in Australian Elections

Richard Buckland¹, Vanessa Teague², and Roland Wen¹

¹ School of Computer Science and Engineering,
The University of New South Wales,
Sydney, Australia

`{richardb,rolandw}@cse.unsw.edu.au`

² Department of Computer Science and Software Engineering,
The University of Melbourne,
Melbourne, Australia
`vjteague@unimelb.edu.au`

Abstract. Research on mitigating vulnerabilities in electronic elections has focused mainly on developing cryptographic voting and counting schemes that satisfy strong mathematical requirements. However many practical problems with e-election systems in general cannot be solved by cryptology. In this paper we consider some of these practical problems by examining deficiencies that are common to the many e-election systems currently used in Australia, including but not limited to e-voting and e-counting systems. We identify poor practices in the commissioning, development, operation and scrutiny of these systems, and we then make recommendations for improving practice. We argue that best practice guidelines for e-election systems need to be explicitly articulated and should include four key elements: failure-critical engineering, risk assessment, a culture of audit and strong transparency.

Keywords: Best practice, electronic elections, e-voting, failure-critical engineering, strong transparency.

1 Introduction

Australia has a long tradition of trustworthy election conduct. The manual execution of elections in Australia is professional, carefully performed and open to public scrutiny. However over the past decade much of the conduct of Australian elections has moved from the manual into the electronic realm, without maintaining the level of quality and transparency achieved with paper-based elections. Election administrators throughout Australia (and the rest of the world) will have to decide on the appropriate way to extend their tradition of election quality and transparency to new technologies, or risk eroding election integrity and public confidence. Our aim in this paper is to identify some shortcomings in current practice and offer suggestions on how to make electronic election processes more secure, reliable and transparent.

E-election systems (that is, any IT system used for elections) in Australia date back to the implementation of the world's first electronic electoral roll in 1967 [27][28]. Since then a vast array of e-election systems has been adopted for almost all aspects of election conduct, and Australian elections have become heavily dependent on these systems. Although publicly available information on Australia's e-election systems remains scant, the little that is available reveals systemic problems. It is apparent that e-election systems are commissioned, developed, operated and scrutinised according to standard industry practices for *commercial* IT systems. These practices are entirely inadequate for *failure-critical* e-election systems, where the operation of the systems is infrequent, intensive, based on secrecy, and where mistakes are not publicly visible and often not even privately evident. The risk these factors pose is of course compounded by the considerable financial incentive for malicious agents to fraudulently manipulate system vulnerabilities to affect election outcomes. These shortcomings in IT practices are most evident in e-voting systems, which have the most stringent requirements, but are equally detrimental to the quality of all other e-election systems.

In this paper we shed light on some of the issues that have been made public and discuss how to address the problems that cause them. The focus of this paper is the practical issues that are common to all e-election systems. Our coverage does not include cryptographic protocols — the fact that Australia does not use cryptographically verifiable e-voting and e-counting schemes is a separate concern. The measures we propose are intended to be used in addition to appropriate cryptographic techniques.

The contribution of this paper is two-fold. First, we gather together in one place the publicly available information on recent incidents in e-election systems deployed in Australia. This is important as it permits analysis and discussion of the common underlying systemic causes, rather than allowing the problems to be treated as a series of independent and isolated incidents. Second, we consider best practice for e-election systems and identify four essential elements. Our intention is to propose a reasonable starting point for developing best practice guidelines for e-election systems.

The structure of the paper is as follows. We begin with an overview of the e-election systems used at present in Australia. Then we examine the four proposed elements of best practice for e-election systems in turn: failure-critical engineering, risk assessment, a culture of audit, and strong transparency. For each element we identify current problems and the steps that are necessary to address these problems.

2 E-election Systems in Australia

Each public election in Australia is administered by a centralised independent electoral commission. The Australian Electoral Commission manages federal elections, and state electoral commissions manage state elections and most local government elections. Electoral commissions employ permanent, full-time

election officials and are responsible for all aspects of electoral administration. This includes conducting elections, reporting on election irregularities, enrolling voters, registering candidates and political parties, redrawing electorate boundaries, educating voters and monitoring political donations.

This centralised approach contrasts with the predominantly decentralised arrangement in the US and most European countries, where the responsibility for conducting elections is typically delegated to the level of local government. Centralisation provides opportunities for economies of scale and professional election administration, but also increases the potential impact of flaws in the e-election systems used.

To help perform their numerous duties and to improve access to the democratic process for voters, each electoral commission has developed its own suite of e-election systems for e-voting, e-counting, electoral roll management and general election administration. This section describes a number of these systems. We focus on those of the largest electoral commissions, namely the Australian Electoral Commission (AEC), the New South Wales Electoral Commission (NSWEC) and the Victorian Electoral Commission (VEC), as well as the Australian Capital Territory Electoral Commission (ACTEC), which was the first to introduce e-voting in Australia.

2.1 E-voting Systems

E-voting systems are attractive in Australia because their flexibility affords high degrees of accessibility and usability, which makes them well-suited to situations where current voting arrangements are inadequate. This is important in Australia, where very strong emphasis is placed on participation in elections. Indeed voting is compulsory. Consequently electoral commissions provide an unusually wide array of voting arrangements to cater for the diverse circumstances of voters. E-voting is becoming more popular as an additional voting option. To date several systems have been trialled or permanently adopted with the purpose of providing an alternative to paper-based voting for voters with visual impairment, voters from a non-English speaking background, and voters living in remote areas or located interstate or overseas.

In 2001 the ACTEC trialled a voting machine system called the Electronic Voting and Counting System (EVACS), which was developed by Software Improvements [3]. EVACS is now used on a permanent basis in major polling places. Each EVACS voting machine displays instructions in a choice of languages, and certain machines designed for visually impaired voters have special facilities including large screens, headphones and audio instructions. In 2007 the AEC conducted a trial for visually impaired voters using EVACS based voting machines [8], but the system was abandoned because of the excessive cost.

The VEC adopted Scytal's Pnyx.DRE voting machine system for visually impaired voters in 2006 [39] and rolled out the system on a larger scale in 2010. All machines have features for non-English speaking voters and visually impaired voters.

Remote e-voting systems remain less common in Australia but have recently garnered increased interest. The AEC conducted a remote e-voting trial in the 2007 Federal Election using the eLect remote voting system by EveryoneCounts [9]. This was for Australian Defence Force personnel deployed overseas and permitted voting only on designated computers connected to a secure private network. The system was later abandoned again due to cost factors.

Most recently the NSWEC has developed a modified version of eLect, called iVote, for large-scale Internet voting during the 2011 NSW State Election. Initially the iVote system was only intended for visually impaired voters and voters living in remote areas. Shortly before the election the scope was substantially expanded to include interstate and overseas voters, as well as voters with *any* disability, including for example poor literacy skills.

2.2 E-counting Systems

E-counting systems have been used in Australia since the late 1980s for the single transferable vote (STV), which is a preferential system for proportional representation. All upper houses of parliament, some lower houses and many local governments are elected with STV. In most cases the votes are counted electronically. This is because the STV counting procedures are sufficiently complex that manual counting is infeasible in large-scale elections.

As multiple variants of STV are used throughout Australia, each electoral commission has its own e-counting system. These systems perform the vote counting and generate detailed statistical reports, many of which are published on electoral commission websites. In response to the desire for enhanced functionality and frequent changes to the STV counting rules, e-counting systems are constantly upgraded or redeveloped. For example the NSWEC has developed at least five new e-counting systems over the last 20 years [29,31].

The e-counting systems also require data capture systems to convert votes from paper ballots into electronic form. Since it is very difficult to ensure the accuracy of automated data capture for preferential ballot papers, almost all the data capture systems currently in use are for manual data entry. The data capture systems provide extensive reporting functions to enable thorough verification of the electronic data against the paper ballots. In the event that e-voting is permitted, the electronic ballots are printed out and then manually entered along with the paper ballots.

An exception is elections in the ACT, where electronic ballot data from the e-voting system is uploaded directly into the e-counting system. In 2008 the ACTEC began using an intelligent character recognition system to scan all paper ballots, but this still involves intensive manual verification [4].

2.3 E-election Systems for Electoral Roll Management

Australia has long used e-election systems for electoral roll management. The primary purpose of these systems is to ensure the roll is accurate and complete, and this is vital given that enrolment and voting are compulsory.

The AEC maintains the national roll, which until recently was used by all state electoral commissions under a joint roll agreement. Enrolment information is mainly provided by voters, and so e-election systems are used by AEC staff to process these applications. Since 2004 the AEC has been developing the General Enrolment, Elections Support and Information System (GENESIS) to replace several legacy e-election systems, and the module for adding and updating voter enrolment details was launched in 2010 [10].

There are also efforts to improve convenience for voters when enrolling and updating their enrolment. Since 2010 the AEC has provided SmartForm online enrolment applications (interactive Adobe Acrobat forms), and it is currently developing a custom online enrolment system.

The trend though is to fully automate the enrolment process so that voter interaction is no longer required. This approach has been advocated to improve completeness of the roll because roughly eight percent of eligible voters are not currently enrolled [11]. In 2010 the NSWEC ended the joint roll agreement with the AEC and began using its SmartRoll automatic enrolment system to add eligible voters to the roll according to data collected from other sources. The VEC followed soon after with its own system and other states are contemplating similar moves. But federal legislation prohibits the AEC from adopting automatic enrolment.

However automated data collection is nothing new. To enhance roll accuracy, electoral commissions have continually been developing ever more sophisticated systems for data collection and data matching. In addition to basic information such as name, address and date of birth, a wide variety of secondary personal information is collected from numerous sources to facilitate data matching, and this can include occupation, previous addresses, phone numbers, drivers licence numbers, tax file numbers and scanned documents containing signatures.

Electronic systems are also used to extract and distribute electoral roll data, for instance to print certified lists of the voters in each electorate for roll marking during elections. In most elections certified voter lists are printed on optical answer sheets, which are later scanned and then uploaded to electronic reporting systems to check for multiple or non voting. Each polling place has copies of the certified list only for the electorate in which it is located. This leads to difficulties in identifying voters for absent voting, where voters attend polling places outside their own electorate.

To address this problem the NSWEC developed the iRoll system in 2007 [29]. This stores the certified lists for all electorates in the state on PDAs or laptops at every polling place, and thus enables polling officials to verify the enrolment details for absent voters. The ACTEC has adopted an enhanced version of iRoll to mark all voters off the roll directly through PDAs [4].

2.4 E-election Systems for General Election Administration

Many e-election systems have been developed in Australia for a wide range of general election administration tasks. Reporting systems extract information from other systems to generate data for purposes such as verifying election

integrity (for instance by tracking ballot boxes and detecting multiple voting) and internal research to identify problems and plan for future elections. Logistical systems are used for operational issues including managing candidate nominations, ballot draws, polling places, polling staff, postal votes, ballot papers, ballot boxes and manual vote counting.

While most of the e-election systems are for internal use, online systems are becoming more common. Voters can now verify their enrolment details online. For the 2010 Federal Election the AEC deployed two online systems: the Online Recruitment System for the public to apply for temporary employment as polling officials, and the Checkpoint system for training polling officials [10].

Online systems have also been used to promote open and broad consultation. A good example is the procedure for electorate redistributions, where electoral commissions periodically redraw electorate boundaries to reflect population changes. To prevent gerrymandering, redistributions involve public inquiries that take place in multiple stages, and any group or individual can submit comments and objections. These inquiries are now conducted online and all the documents and submissions are located on electoral commission websites.

The large number of e-election systems in use and the desire to provide online interaction with these systems for both voters and electoral commission staff has motivated the need to integrate these systems and expand interoperability. Indeed this is the reason behind the AEC's ongoing GENESIS project. Likewise the NSWEC implemented a web-based Election Management Application for similar purposes, and this was deployed in 2006 [29].

3 Failure-Critical Engineering

E-election systems are failure critical. Failures in any of these systems could have extensive and catastrophic consequences for overall election security and integrity, as well as undermining public confidence in the electoral process. To maintain the quality and trustworthiness of elections, rigorous failure-critical engineering practices need to be followed to ensure that e-election systems are of the highest standard.

3.1 Current Problems

In Australia e-election systems are currently treated as regular commercial IT systems instead of failure-critical systems. This approach has resulted in a large number of systems failing during crucial periods. The problems are most evident throughout the software development process, where “the gap between the best software engineering practice and the average practice is very wide — perhaps wider than in any other engineering discipline” [19].

For example, the NSWEC iVote system suffered multiple failures during the 2011 NSW State Election [33]. The most critical incident was the corruption of votes by problems with the client-side JavaScript.

The iVote user interface required voters to enter their preference rankings in order, starting from 1. This was done by selecting a candidate and then pressing

the letter ‘N’ to allocate the next preference ranking to this candidate. However the NSWEC discovered 43 ballots where the letter ‘N’ was stored as a preference in place of some of the numerical preferences. This shows how a single, minor software bug has the power to corrupt votes without being detected by voters, and raises doubts over whether *any* vote at all was cast as the voter intended.

Furthermore the iVote back end had inadequate input validation and error reporting functions to identify invalid votes; the vote corruption was discovered only when election officials noticed a discrepancy between the number of electronic ballots cast and the number of ballots printed for counting.

This incident demonstrates failings in the software design, implementation and testing. In addition we observed that iVote compromised core requirements. Notably iVote was supposed to provide audio instructions, like all other Australian e-voting systems for visually impaired voters. However this requirement was dropped.

Critical incidents have occurred previously in NSW elections. During the 2003 NSW State Election the e-counting system experienced irrecoverable failures, and this led to delays in publishing the final result [31]. The problems were in part due to a lack of input validation and error reporting functions in the vote data entry system. Additionally there were separate problems with database configuration and maintenance. The e-counting system also suffered from less critical bugs such as inexplicable error messages.

These basic defects were not discovered during extensive end-to-end functional testing. The heavy reliance on such high level testing techniques appears to be commonplace. Similar functional testing was performed on the ACTEC’s EVACS counting module [3], which likewise experienced failures during the 2001 ACT Election [25].

Of particular concern is that end-to-end testing is used to verify the correctness of the counting algorithm implementation in e-counting systems. The value of black box testing methods for this purpose is highly questionable because STV counting algorithms are extraordinarily complicated and prone to subtle implementation flaws. Indeed there are several instances in Australia where even the legislation specifying the STV counting procedures contains omissions and/or internal conflicts, so that it is mathematically impossible to count the votes according to the prescribed algorithms [40]. The fact that such legislative irregularities were not discovered when the e-counting systems were being specified, developed and tested reflects shortcomings in the testing as well as the specification of these systems.

The lack of testing rigour is widespread. Critical failures have occurred with the iRoll system for marking voters off the roll using PDAs. In the 2007 Tasmanian State Election, iRoll experienced data corruption when uploading the details from the PDAs, which resulted in the inability to determine whether 500 voters had been marked off the roll [37]. To address this problem the ACTEC enhanced iRoll so that the PDAs immediately backed up the data to a master PDA in each polling place via Bluetooth. The backup system failed during the

2008 ACT Election [4]. Fortunately no PDA problems arose on that occasion and no data was lost.

In the 2010 Federal Election AEC staff reported extensive failures with the enrolment processing module of GENESIS, the Online Recruitment System and the Checkpoint online training system [20]. The problems included poor performance (partly due to insufficiently powerful server hardware), poor usability, missing functionality and glitches such as freezes, crashes and outages. This created numerous and unprecedented challenges for AEC staff and required temporary workarounds to counter issues with the systems.

The testing processes for these projects had serious shortcomings. AEC staff identified problems with GENESIS in early user testing but these were dismissed, and the Online Recruitment System did not undergo any live testing prior to launch despite concerns raised by staff [20].

Many of the problems with GENESIS have stemmed from inadequate requirements analysis. An audit of the AEC's conduct of the 2007 Federal Election by the Australian Auditor-General found that a poor understanding of the requirements for GENESIS contributed to a delay of over three years to the completion date so far (now estimated to be the end of 2014) and a cost blowout from \$27 million originally to between \$56 and \$60 million now expected [2].

There are also indications of problems with code design and implementation. The public source code for the ACTEC's EVACS was separately reviewed by researchers from the Australian National University [1] and the University of California, Davis [23]. Their studies found unclear design, large amounts of duplicate code, complex control flow, minimal error checking, memory leaks and hard-coded values. It seems fair to expect similar software risks in other e-election systems in Australia. EVACS has at least had the advantage of allowing open scrutiny, and as a result some of these problems have since been fixed. Given that no source code for any other systems has been publicly disclosed, the quality of the code for these systems is likely to be worse.

One of the main reasons behind all these failures is that electoral commissions are not equipped with the requisite expertise and resources to establish and implement failure-critical engineering practices. Although the NSWEC acknowledged this when assessing the failures of its e-counting system [31], the same practices still persist in Australia.

At present it is commonplace for electoral commissions to engage consultants to manage e-election projects, and to outsource part or all of the development, evaluation and operation of e-election systems to private contractors or vendors. However these external parties are general IT practitioners rather than specialists in failure-critical systems. Consequently electoral commissions have very limited capabilities to ensure the quality of these systems.

Outsourcing in this manner has had catastrophic consequences for Dutch e-voting systems [32]. The problem is even more concerning in Australia because of the heavy usage of and reliance on e-election systems for almost all facets of election conduct.

3.2 Towards Best Practice

E-election systems must be commissioned and managed as failure-critical systems. The inherent complexity of IT systems makes it very easy to inadvertently or deliberately introduce defects into a system during the development process, and at the same time makes it notoriously difficult to detect and eliminate all the defects. Likewise the operation and use of IT systems is highly vulnerable to human error and malicious activity. Best practice for e-election systems must adhere to engineering practices that are specifically designed to mitigate these problems from the outset and to ensure security, reliability and usability.

In more mature domains, failure-critical engineering has proven to be successful in developing the highest quality systems such as avionics systems, medical equipment and even computer hardware. For example comprehensive and systematic techniques (such as formal specification and verification) are employed to minimise the introduction of defects and produce objective evidence that the systems are secure and reliable. Robust safeguards are built into the systems so that potential failures can be detected, reported and handled gracefully, rather than causing a total and possibly unnoticed collapse.

Best practice guidelines for the engineering of e-election systems can adopt many of these well-established practices and principles. Once these guidelines have been prescribed, the practices need to be overseen and implemented by a diverse range of suitably qualified experts with extensive training and experience in the necessary areas, including software engineering, failure-critical engineering and security engineering.

4 Risk Assessment

Risk assessment lays the foundation for sensibly managing and appropriately dealing with the risks involved in the development, operation and use of e-election systems. E-election systems have unique threats and vulnerabilities, with serious irreversible and far-reaching implications that may even extend beyond the electoral realm and into the public sphere. It is essential to enumerate all the risks and consider their potential impact. This permits well-informed management decisions to be made about whether or not to commission a system in the first place, and then if so what development processes are best adopted and what further technical and procedural safeguards are needed.

4.1 Current Problems

Current e-election system risk assessments in Australia suffer from multiple inadequacies, most notably in their narrow scope and lack of rigour. This has led to poor decision making that has exposed elections to high risks, and in a number of instances these risks have been realised.

There has been a consistent failure to properly assess the risks in software development, often resulting in foreseeable IT problems. In particular it has become the norm for new e-election systems to be developed on a tight schedule and

then to be deployed at the most crucial point of the electoral cycle. For example development started roughly *six months* before the election for the NSWEC's iVote, the AEC's trial e-voting systems and the ACTEC's EVACS. Given that this leaves little margin for error and that IT projects have the propensity to be delayed, such a short time frame poses a serious risk of compromising the quality of these systems in order to meet critical deadlines. In the case of iVote, the auditor noted that this resulted in "incomplete documentation, restricted test case formulation and compressed testing activities" [33].

Long term projects have also experienced the same problem. Many of the deficiencies with the AEC's systems during the 2010 Federal Election were aggravated because there was insufficient time scheduled to perform live testing. Furthermore the decision was made to launch some of these systems even after problems were identified.

Accurate risk assessments are vital for security. Risk profiles can change subtly and unexpectedly, yet at present the risks are not reassessed on an ongoing basis. This is especially problematic with function creep.

A recent case is the expansion of the NSWEC iVote system to include interstate and overseas voters. As a result almost 50 000 votes were cast over the Internet, which was ten times the number anticipated and predominantly comprised votes from interstate and overseas. This drastically changed the risk profile from a controlled small-scale trial, where problems would likely have a reasonably minor impact, to an uncontrolled large-scale event, where problems could have a major impact.

In the tightly contested seat of Balmain, the winning margin was around 100 votes. Over 900 votes for this seat were cast with iVote, and so relatively minor problems could have affected the outcome. In close elections similar scope changes could have implications for the integrity of the overall election result.

Standard security vulnerabilities are also frequently overlooked or underestimated. Many e-election systems have been developed without protection against even basic attacks. For instance the ACTEC's EVACS uses voting clients that send unencrypted votes to a ballot box server within the polling place via a local network [23]. Hence vote privacy and integrity could easily be compromised by even an unsophisticated attacker who gains access to the network. Also the voting clients do not store an independent audit trail of the votes cast, and so the ballot server presents a single point of failure against malicious attack or hardware failure.

In a similar way the use of highly insecure Bluetooth wireless technology in the ACTEC's iRoll backup system provides a vector for an attacker to gain unauthorised access to the PDAs containing the certified voter lists. An attacker could potentially violate the privacy and integrity of the certified lists, for instance to facilitate multiple voting by 'unmarking' voters from the certified lists. The nature of Bluetooth wireless communication means this attack could happen anonymously and at a distance. Again the attacker need not be particularly sophisticated or well resourced to carry out such a multiple voting attack.

The combined usage of multiple e-election systems has introduced new risks that have not been considered. For instance the simultaneous use of iRoll and EVACS by the ACTEC has created the potential for voter privacy to be violated through exploiting electronic metadata, in this case timestamps. The ACTEC's version of iRoll tracks voter flow in polling places by storing timestamps of when the voters were marked off the roll [4]. Also EVACS electronic vote records include timestamps of when the votes were cast. Cross-referencing iRoll timestamps with vote timestamps could then reveal potential matches between voters and votes. This technique would be effective when there are large gaps between timestamps, which will be the case during the three week pre-poll period and at quiet times on election day, particularly in smaller polling places. This vulnerability also highlights the subtle dangers of function creep, in this instance through minor feature enhancements.

The possible impact of individual e-election systems on overall election quality tends to be overlooked. Many systems are not even recognised as being failure critical. This is compounded by the ongoing trend to integrate all e-election systems. Systems deemed critical are placed at greater risk because an outside attacker could exploit a vulnerability in a 'non-critical' system to gain inside access. For example the AEC assumed its e-counting system was secure as the system was (in theory) only operated on a standalone machine isolated from the network, when in fact the system was designed with the capability to operate in a networked environment [5]. Such intrinsically insecure systems may be permanently exposed to higher risk because there may be limited scope to later harden them against attack.

There has also been a failure to consider the threats that e-election systems may pose outside elections. A prominent example is the privacy implications of e-election systems for the electoral roll such as the NSWEC's SmartRoll automatic enrolment system. These continue the expansion of function creep in electronic electoral roll systems, where a series of seemingly minor and insignificant changes has ended up having a large and unanticipated compound effect, not only on elections but also potentially on the everyday lives of voters.

The increasingly large volume and variety of data collected for the electoral roll has not only amplified the scale of the risks of violating the privacy of personal voter information, but has also changed the very nature of the risks. The secondary personal information now stored on voters is highly sensitive, and so leaking roll data can have extremely harmful consequences including identity fraud. Moreover the introduction of distributed systems such as iRoll and the growth in authorised third party access to the roll have increased the risk for roll data to be leaked through loss or theft.

4.2 Towards Best Practice

Risk assessments must be performed on all e-election systems, and these assessments must be comprehensive, ongoing and involve broad consultation. An overarching framework is necessary to set out the appropriate methodologies for conducting accurate risk assessments.

This framework should specify a systematic approach to identifying and gauging the full range and extent of the constantly evolving risks, especially in relation to low-probability, high-impact events. It needs to ensure broad consideration of the technologies, procedures and policies associated with e-election systems. Furthermore it must provide a holistic examination of the risks each system can pose to other systems and the entire election process, rather than dealing with the risks of each system in isolation.

5 A Culture of Audit

Rigorous independent audits are crucial for assuring the quality of e-election systems by critically reviewing all aspects of the systems. These audits cannot be a last-minute and secondary concern whose role is to tick the box or otherwise on an already delivered system. Instead a culture of audit needs to be adopted throughout the design, development, operation and post-mortem of e-election systems to develop high quality and an assurance of that high quality.

5.1 Current Problems

Audits in Australia are not conducted with sufficient time or expertise, and this has allowed problems with e-election systems and engineering practices to be overlooked. In many instances audits are not even performed. This was the case for all of the AEC's new systems and the NSWEC's e-counting system. Furthermore the Australian Auditor-General commented on the poor documentation of the AEC's systems and development processes [2], which would make it very difficult to perform audits if desired.

Even for the most critical systems, audits are often treated as an afterthought. In the case of the NSWEC iVote system, the feasibility study [30] originally scheduled less than eight days in total for conducting the audit and addressing the findings, with the voting period commencing just 10 days later! It would be unreasonable to expect that major flaws with such a complex system could be discovered and fixed in such a short time frame.

Note that voter registration for iVote was scheduled to open two weeks before the audit was due to be completed. Thus in the face of adverse findings, the NSWEC would have faced a very difficult decision between proceeding with using a vulnerable system in a failure-critical environment, or abandoning the system and potentially disfranchising 50 000 voters who planned to use iVote.

The iVote audit reports revealed that critical issues still remained one week prior to going live [34] and several vulnerabilities were not addressed in time [33]. There is no publicly available information on the nature or gravity of these vulnerabilities, but it appears that the final decision to launch iVote was made with knowledge that the system had significant security and quality shortcomings.

We observed that iVote was highly vulnerable to malicious hacking and to automated malware because there was no client-side encryption of the votes (aside from TLS/SSL encryption for HTTPS), and so voters' PCs sent, and

the NSWEC's voting servers received, the votes in plaintext form. This obvious security vulnerability was evident through our brief inspection of the live system.

The superficial nature of audits appears to be common. The ACTEC's EVACS counting module was certified by an independent auditor despite having serious defects that caused failures during the 2001 ACT Election [25]. Also the auditor failed to identify numerous other defects that could have caused incorrect election results and segmentation faults, and may have permitted penetration by malware. Many of these were obvious and elementary software defects that were later identified using standard testing methods and very simple test cases by the Australian National University review of EVACS [1].

Threats and vulnerabilities are consistently overlooked in security audits. For instance in examining the eLect source code for the AEC's remote e-voting trial system, the auditor only considered the possibility of malicious source code rather than inadvertent faults that could create security vulnerabilities [21]. This had longer term consequences extending beyond the AEC trial because the NSWEC iVote system was subsequently based on eLect.

Even when well-known security vulnerabilities are identified, their significance is not always properly understood by non-expert auditors. For example the e-voting system for the 2010 Victorian State Election used a weak, non-standard method for seeding pseudorandom number generators for some cryptographic keys [22]. The auditor (and, unsurprisingly, the vendor) dismissed this vulnerability as being negligible [18,36], in spite of the fact that it belongs to a well-known class of security vulnerabilities. Such weaknesses have been famously exploited in other systems, for instance the Netscape SSL attack [26].

5.2 Towards Best Practice

Comprehensive and continuous expert audits must be integrated into e-election systems practice, as they are for other failure-critical systems. Given the large number and variety of complex issues that all need to be examined, this approach to auditing is necessary to assure the high quality of e-election systems.

A culture of audit provides an essential layer of defence for directly identifying problems with the technology and operational procedures, as well as uncovering systematic weaknesses in the engineering and risk assessment practices that are likely to cause or overlook problems. It prevents failures and vulnerabilities by discovering and rectifying problems early on, rather than simply detecting them when it is too late.

6 Strong Transparency

Strong transparency is a fundamental feature of trustworthy elections. It promotes public understanding and involvement in the scrutiny process. By enabling such broad scrutiny of all aspects of elections, transparency also assures and enhances election integrity. This unique requirement for the highest level of transparency has even greater importance for e-election systems, which by nature are incredibly hard to understand and scrutinise.

6.1 Current Problems

So far the use of e-election systems in Australia has substantially eroded election transparency. The mere usage of these systems has managed to obscure many formerly manual election processes. There is minimal information on what e-elections systems are used by electoral commissions, how these systems operate and what failures occurred during elections.

For example most of the public details regarding problems with the AEC's e-election systems in the 2010 Federal Election only came to light through a parliamentary submission by the union that represents AEC staff [20]. It seems likely that many other problems were not reported at all.

Even for the most critical failures, few or no details are made public. This is particularly concerning in the case of the vote corruption in iVote, where the NSWEC determined some of the corrupted votes to be invalid, whilst also determining the intent of others [33]. The NSWEC has no intention to publish any of the documentation relating to the iVote incidents, why they occurred and how they were resolved.

In many instances the outsourcing of e-election systems has created barriers to transparency because of the intellectual property issues. Private vendors are reluctant to make source code or any other details of their systems publicly available. Even independent experts and auditors engaged in evaluating the security and reliability of an e-election system are usually forced to sign non-disclosure agreements with the vendor in order to gain access to source code. These agreements typically prohibit any comments from being made about the system without the vendor's prior approval, and naturally this can severely limit the public disclosure of adverse findings.

No technical documentation on any Australian e-election systems is publicly available. Audit reports and other high level reports have been published for the e-voting systems used in the 2007 Federal Election [7,9,6,8,16,17], the 2010 Victorian State Election [18,22,36] and the 2011 NSW State Election [33,34]. However most contain minimal technical detail and some refer to unpublished primary documents, including security and code reviews. In addition these reports were all released well after the elections took place (six months later for the Victorian reports).

The only publicly available source code is for the ACTEC's EVACS, but code for some parts of the system has not been disclosed. Other electoral commissions have repeatedly resisted calls to publish source code for their systems. For example a parliamentary inquiry recommended that the VEC should publish the source code for its e-counting system on its website and collect comments and bug reports from the public [35]. But this recommendation was disregarded. Instead the VEC has so far maintained that it is sufficient to have the system independently audited and then to provide electronic ballot data to scrutineers, who can calculate the election result and compare it to the published results [38]. Yet the e-counting system audit report was never published.

Furthermore giving the ballot data to scrutineers does not guarantee that the counting will be thoroughly scrutinised. Political parties may lack the expertise

and resources to develop complex STV software that counts the votes and generates the highly detailed data necessary to verify the official results data.

Moreover revealing ballot data in preferential electoral systems can expose voters to coercion through signature attacks [24], which are easy to carry out given the electronic ballot data. In an attempt to allow some verification of the count (though not of the ballot data), the AEC and ACTEC publish ballot data on their websites. It remains unclear how best to provide meaningful verification of electronic STV counting while mitigating the risk of large-scale voter coercion.

A lack of transparency has in certain instances created a great deal of voter confusion. This has been the case with automatic enrolment in NSW and Victoria. For example the NSWEC SmartRoll system notifies voters of automatically added enrolments and updated addresses. But few of these voters realise that this is only for the state roll and that they must still manually update their details for the federal roll [10]. Consequently the NSW state electoral roll has diverged substantially from the federal roll for NSW.

Non-transparency of e-election systems has also had an impact outside of elections. For example the use of highly sophisticated systems to collect vast volumes of voter data for the electoral roll has created a situation where the public has little idea of what types of personal information are gathered, which sources they come from, when the data is collected and to which third parties they are subsequently provided. Although individuals can inspect roll information that is available to the general public (primarily names and addresses), they do not have any means to verify or identify the secondary information collected and maintained on themselves. Concerns over privacy violations stemming from the lack of transparency over roll data have been raised on multiple occasions, for instance by the Australian Privacy Commissioner [12,13,14,15] and Australian Auditor-General [2], but the issues have yet to be properly addressed.

This privacy risk highlights the dilemma where on the one hand electoral commissions are under pressure to develop e-election systems to improve the democratic process; but on the other these systems may unexpectedly come into conflict with the public interest. The absence of transparency has hampered public debate over whether the risks and trade-offs are acceptable.

6.2 Towards Best Practice

E-election systems must be strongly transparent. The existing commitment and dedication to the transparency of manual election processes need to be applied to e-election systems as well. While the underlying principles of transparency remain the same, the challenge is that the complexity and capabilities of electronic systems make them inherently obfuscated compared to manual systems.

As a result concerted effort is necessary to make e-election systems transparent *by design*, so that there is openness as well as supporting material for this openness. The highest level of disclosure is needed to reveal the full details of the systems well in advance of an election. This includes source code and technical documentation, user and training manuals, and reports for the development processes, operational processes, risk assessments, reviews and audits. For this to be

possible the engineering, risk assessment and auditing processes must generate high quality documentation that is promptly released for public inspection.

Only through strong transparency can the public gain a clearer understanding of e-election systems and participate in the scrutiny process. This will then improve the quality and trustworthiness of e-election systems by allowing the benefits and problems with these systems to be accurately identified, and enabling well-informed public discussion about whether the right decisions are made in developing and using the systems.

7 Conclusion

In this paper we have set out four elements of best practice in the development, deployment and scrutiny of e-election systems. By encouraging the development and adherence to a process of best practice, our hope is to sustain the current high level of quality and public confidence in the conduct of Australian elections as they increasingly move away from manual systems over the coming decade.

So far Australia has embraced the benefits of e-election systems without sufficient consideration of how they fundamentally change the nature of elections and how to address these issues. The regularity and scale with which serious problems occur in e-election systems and their development is a clear and present danger to the security of Australian elections.

These problems demonstrate the need for cultural change to establish and adopt best practices that guarantee rigour in the engineering, risk assessment and auditing processes for e-election systems, and that provide strong transparency of these technologies and processes. This is becoming increasingly urgent as many systems are currently being rapidly (re)developed, in particular for e-voting.

Best practice is still essential when good cryptographic solutions are adopted. Most of Australia's e-election systems lie outside the scope of advanced cryptographic protocols, which are only for e-voting and e-counting. As a result the overall election process cannot be protected by cryptographic means alone. Even with strong cryptographic verifiability through protocols such as Helios, some vote corruption would still go unnoticed by many voters. Furthermore there is no satisfactory way to recover from catastrophic failures when (or if!) detected by diligent voters. In a sense cryptographic verifiability is a last line of defence to help *detect* failures and assure election integrity if that was indeed maintained. Additional layers of defence are equally important to help *prevent* failures.

Best practices for e-election systems do not need radical new or prohibitively expensive methods, but can be largely based on well-established best practices for IT, failure-critical systems and traditional paper-based elections. Common sense dictates that these very straightforward and proven practices should be applied to e-election systems. Yet they are not followed in Australia and it seems likely that the situation is similar elsewhere, considering the controversies over the general poor quality of e-voting systems worldwide. This motivates the broader need to establish guidelines that explicitly stipulate what constitutes best practice.

References

1. Abate, P., Dawson, J., Goré, R., Gray, M., Norrish, M., Slater, A.: Formal Methods Applied To Electronic Voting Systems. Tech. rep., College of Engineering and Computer Science, The Australian National University (2004)
2. Australian Auditor-General: The Australian Electoral Commission's Preparation for and Conduct of the 2007 Federal General Election. Audit Report No. 28 2009–2010, Australian National Audit Office (2010)
3. Australian Capital Territory Electoral Commission: The 2001 ACT Legislative Assembly Election: Electronic Voting and Counting System Review (2002)
4. Australian Capital Territory Electoral Commission: Report on the ACT Legislative Assembly Election 2008 (2009)
5. Australian Electoral Commission: Submission 181 (supplementary), Inquiry into the 2001 Federal Election. Joint Standing Committee on Electoral Matters, Parliament of Australia (2003)
6. Australian Electoral Commission: Final Evaluation Report: Evaluation of the Electronic Voting Trial for Blind and Sight Impaired Electors at the 2007 Federal Election (2008)
7. Australian Electoral Commission: Final Evaluation Report: Evaluation of the Remote Electronic Voting Trial for Overseas Based ADF Personnel Electors at the 2007 Federal Election (2008)
8. Australian Electoral Commission: Report into Electronically Assisted Voting at the 2007 Federal Election for Electors who are Blind or Have Low Vision (2008)
9. Australian Electoral Commission: Report into Remote Electronic Voting at the 2007 Federal Election for Overseas Australian Defence Force Personnel (2008)
10. Australian Electoral Commission: Submission 87, Inquiry into the 2010 Federal Election. Joint Standing Committee on Electoral Matters, Parliament of Australia (2011)
11. Australian Government: Strengthening Australia's Democracy. Electoral Reform Green Paper (2009)
12. Australian Privacy Commissioner: Submission 42, Inquiry into the Integrity of the Electoral Roll. Joint Standing Committee on Electoral Matters, Parliament of Australia (2000)
13. Australian Privacy Commissioner: Submission 154, Inquiry into the 2001 Federal Election. Joint Standing Committee on Electoral Matters, Parliament of Australia (2002)
14. Australian Privacy Commissioner: Submission 164 (supplementary), Inquiry into the 2001 Federal Election. Joint Standing Committee on Electoral Matters, Parliament of Australia (2002)
15. Australian Privacy Commissioner: Submission 172 (supplementary), Inquiry into the 2001 Federal Election. Joint Standing Committee on Electoral Matters, Parliament of Australia (2002)
16. BMM Australia: Audit and Certification of a Remote Electronic Voting System for Overseas Australian Defence Force Personnel. Australian Electoral Commission (2007)
17. BMM Australia: Audit of AEC's Electronic Voting Machine for Blind and Vision Impaired Voters. Australian Electoral Commission (2007)
18. BMM Australia: Electronically Assisted Voting Audit. Victorian Electoral Commission (2010)

19. Brooks Jr., F.P.: No Silver Bullet - Essence and Accidents of Software Engineering. *IEEE Computer* 20(4), 10–19 (1987)
20. Community and Public Sector Union: Submission 95, Inquiry into the 2010 Federal Election. Joint Standing Committee on Electoral Matters, Parliament of Australia (2011)
21. Computing Research and Education Association of Australasia: Submission 116.1 (supplementary), Inquiry into the 2007 Federal Election. Joint Standing Committee on Electoral Matters, Parliament of Australia (2008)
22. Computing Research and Education Association of Australasia: Report on the VEC-Scytl Electronic Voting System for the 2010 Victorian Election. Victorian Electoral Commission (2010)
23. Das, A., Niu, Y., Stegers, T.: Security Analysis of the eVACS Open-Source Voting System. Tech. rep., Department of Computer Science. University of California, Davis (2005)
24. Di Cosmo, R.: On Privacy and Anonymity in Electronic and Non Electronic Voting: the Ballot-As-Signature Attack. HAL Open Archive Document hal-00142440, version 2 (2007)
25. Downie, G.: Libs set to take third seat in Molonglo. *The Canberra Times* (October 2001)
26. Goldberg, I., Wagner, D.: Randomness and the Netscape Browser. *Dr. Dobb's Journal* (1996)
27. Joint Standing Committee on Electoral Matters, Parliament of Australia: The Conduct of Elections: New Boundaries for Cooperation (1992)
28. Macilwain, M.: History of the State Electoral Office, 1907-2007. South Australian Electoral Office (2007)
29. New South Wales Electoral Commission: Annual Report 2006-2007 (2007)
30. New South Wales Electoral Commission: Report on the Feasibility of Providing iVote Remote Electronic Voting System (2010)
31. New South Wales Electoral Office: Submission 10, Inquiry into the Administration of the 2003 NSW Election. Joint Standing Committee on Electoral Matters, Parliament of New South Wales (2005)
32. Oostveen, A.M.: Outsourcing Democracy: Losing Control of E-Voting in the Netherlands. *Policy & Internet* 2(4), 201–220 (2010)
33. PricewaterhouseCoopers: iVote Post Implementation Report. New South Wales Electoral Commission (2011)
34. PricewaterhouseCoopers: iVote Pre Implementation Report. New South Wales Electoral Commission (2011)
35. Scrutiny of Acts and Regulations Committee, Parliament of Victoria: Final Report on the Inquiry into Electronic Democracy (2005)
36. Scytl: Comments from Scytl on the CORE Report from the Electronic Voting Solution Used in 2010 Victorian Election. Victorian Electoral Commission (2011)
37. Tasmanian Electoral Commission: Annual Report 2006-2007 (2007)
38. Victorian Electoral Commission: Submission 27, Inquiry into Electronic Democracy. Scrutiny of Acts and Regulations Committee, Parliament of Victoria (2005)
39. Victorian Electoral Commission: Report to Parliament on the 2006 Victorian State Election — Submission 20, Inquiry into the 2006 Victorian State Election. Electoral Matters Committee, Parliament of Victoria (2007)
40. Wen, R.: Online Elections in Terra Australis. Ph.D. thesis, School of Computer Science and Engineering, The University of New South Wales (2010)

On the Side-Effects of Introducing E-Voting

James Heather¹, Morgan Llewellyn², Vanessa Teague³, and Roland Wen⁴

¹ Department of Computing, University of Surrey, Guildford, Surrey GU2 7XH, UK
j.heather@surrey.ac.uk

² IMT Lucca

hlllewell@gmail.com

³ Dept. Computer Science and Software Engineering, The University of Melbourne
vjteague@unimelb.edu.au

⁴ The University of New South Wales, Sydney
rolandw@cse.unsw.edu.au

Abstract. The literature abounds with discussions on the relative security merits of various voting systems, and on whether a move towards electronic voting is, from a security perspective, something to be encouraged or discouraged. Little has been said, however, on whether there would be unintended side-effects of changing the voting technology, in terms of the votes cast. Security issues aside, should we expect the introduction of an electronic voting system to affect the results of the election?

This paper attempts to tease out some of the possible effects, by analysing ballot data from the 2008 Australian Capital Territory (ACT) Legislative Assembly Election.

1 Introduction

Significant changes to existing voting technology may produce intended or unintended side-effects on voter behaviour and participation. For instance, in response to the problems with the 2000 Florida presidential election, Florida voting precincts experienced widespread voting technology change from 2000 to 2004. Results of this change suggest that the change in voting technology was responsible for a 90% drop in the residual vote rate (that is, the proportion of ballots that were spoiled) during the 2004 election [HPT⁺10]. Similarly, a voting technology change between 2000 and 2004 is attributed to a 35% drop-off in residual vote rates in Michigan [HPT⁺10]. While debate exists over the significance of the finding, proponents of postal voting often contend this technology increases voter turnout relative to traditional polling station voting.

New voting technologies may also affect voter behaviour by altering perceptions of ballot security and secrecy of the ballot choice. Differences in voter trust, or lack thereof, is often cited as an important consideration when proposing a change in voting technologies [AHT09]. Implementation of a new voting technology without adequate poll worker training may reduce voter evaluations of the voting process [HMP09]. Changes to existing voting technology can also take a

partisan tone if ballot order and turnout are affected. It remains an open question if differences in voting technology affect vote choice and turnout decisions.

Indeed, there are specific instances of election officials implementing a new voting technology in an effort to change voter behaviour. For instance, in the context of Australian elections, a randomized ballot ordering was introduced with the explicit intent of evenly dispersing randomly cast votes across all candidates.

This paper uses data from the 2008 Australian Capital Territory (ACT) Legislative Assembly election to analyze differences in voter behaviour across electronic and paper-based voting technologies. We also draw out several other aspects of voter behaviour from the data.

The results are important in gaining an understanding of how the electoral system can affect the results. This will be of particular interest to researchers who design and build new election technology. For instance, one of the most promising contenders for a secure voting system that could see real world use is Prêt à Voter [RBH⁺09]. Introducing Prêt à Voter into a jurisdiction that currently uses a traditional voting system would require switching to an electronic infrastructure; it would also involve randomizing the order of the names of the candidates on the ballot paper; and it may mean the publication of full (anonymized) ballot data. Would making such changes have unintended side-effects on voter behaviour?


2 ACT Legislative Assembly Elections

The ACT Legislative Assembly is a state-level unicameral parliament with three multi-seat electorates: Brindabella (five seats), Ginninderra (five seats) and Molonglo (seven seats). Members are elected using the Simple Gregory variant of the single transferable vote (STV), which is an electoral system where voters rank the candidates in order of preference, with a minimum of one preference. In a typical election there are 20–30 candidates per electorate, with the major parties being Labor, the Liberals and the Greens.

An unusual feature of ACT Legislative Assembly elections is the use of ‘Robson rotation’ ballots (see Figure 1). Under Robson rotation, the order of the parties is fixed (chosen by a random draw, independently for each electorate), but the order of candidates within each party is rotated in such a way that there are numerous different versions of each ballot paper (60 for Brindabella and Ginninderra, and 420 for Molonglo). The result of the Robson rotation is that, taken over the entire electorate, no single candidate benefits from random voting (such as simply voting for a party’s first n candidates). What is most interesting about Robson rotation is that it was designed to alter voter behaviour; we will discuss this in Section 4.

Also unlike most STV elections in Australia, Legislative Assembly elections do not have group voting tickets, where voters have the additional option of selecting a ticket that corresponds to an ordering of preferences predetermined by groups of political parties and/or independent candidates. Group voting tickets tend to

+

Ballot Paper Election of 5 Members 2008  Legislative Assembly for the Australian Capital Territory

Electorate of Brindabella

Number five boxes from 1 to 5 in the order of your choice

You may then show as many further preferences as you wish by writing numbers from 6 onwards in other boxes

A CANBERRA LIBERALS	B COMMUNITY ALLIANCE	C AUSTRALIAN MOTORIST PARTY	D THE GREENS	E AUSTRALIAN LABOR PARTY
<input type="checkbox"/> Brendan SMYTH	<input type="checkbox"/> James SIZER	<input type="checkbox"/> Brian McLACHLAN	<input type="checkbox"/> Amanda BRESNAN	<input type="checkbox"/> Mick GENTLEMAN
<input type="checkbox"/> David MORGAN	<input type="checkbox"/> Val JEFFERY	<input type="checkbox"/> Ben DOBLE	<input type="checkbox"/> Sue ELLERMAN	<input type="checkbox"/> Joy BURCH
<input type="checkbox"/> Steve PRATT		<input type="checkbox"/> Geoff RAKE		<input type="checkbox"/> Wayne SIEVERS
<input type="checkbox"/> Steve DOSZPOT		<input type="checkbox"/> Burl DOBLE		<input type="checkbox"/> Tracy MACKAY
<input type="checkbox"/> Audrey RAY	SAMPLE	<input type="checkbox"/> Bruce RITCHIE		<input type="checkbox"/> John HARGREAVES

1 Remember, number at least 5 boxes from 1 to 5 in the order of your choice 1

+

Fig. 1. Sample Ballot Paper for Brindabella

discourage expressivity because voters are more inclined simply to choose the convenience of a group ticket instead of ranking the candidates in the preferred order. For example, in Federal STV elections, over 95% of voters typically use group tickets. From the point of view of the data analyst, it is helpful that the ACT does not allow group tickets, because they distort the ballot data by encouraging voters to vote for a quick approximation to their true preferences.

Like all elections in Australia, ACT Legislative Assembly elections provide voters with numerous voting options such as pre-poll at major polling places and postal voting. Voters can also vote at any polling place, even outside their home electorate. The ACT is very progressive compared to other Australian jurisdictions in that e-voting has been available as a voting option in some polling stations for the last 10 years.

For the 2001 Legislative Assembly Election, the ACT Electoral Commission developed the Electronic Voting and Counting System (EVACS). The system was initially used on a trial basis but is now widely used on a permanent basis. The e-voting component of EVACS comprises voting machines that display instructions in a choice of languages, and certain machines designed for visually impaired voters have special facilities including large screens, headphones and audio instructions. EVACS voting machines are available in major polling places and can be used by any voter.

EVACS also has an e-counting module that implements the STV counting procedure, which would be extraordinarily difficult to conduct by hand. Paper ballots are scanned with an intelligent character recognition system and manually verified, then uploaded into EVACS and combined with the electronically cast ballots.

This unique combination of e-voting and e-counting for STV generates extraordinarily useful electronic preferential ballot data. To facilitate statistical analysis, this data contains not only each vote's candidate rankings but also metadata including:

- the Robson rotation ballot positions of the candidates,
- the polling place at which the vote was cast,
- whether the vote was cast electronically or on paper, and
- whether the vote was cast during pre-polling or on election day.

This data is publicly available on the ACT Election Commission website. Our analysis in this paper is drawn from the data for the 2008 Legislative Assembly Election [\[ACT08\]](#).

3 Party Selection and Voting Interface

In some polling places, ACT voters were given a choice as to whether to vote electronically or on paper. We analyze voter choices over the voting technology in order to determine if correlation exists between the content of the vote and the method of casting. Of 84 polling places in the ACT 2008 election, five locations (City, Tuggeranong, Belconnen, Gungahlin, Woden) offered the option of electronic voting, to polling day voters and to pre-poll voters. There were no polling places that mandated electronic voting.

Across the 84 polling sites, only 20% of votes were cast electronically. However, the low rate of electronic voting seems to stem from the fact that only five polling places offered the option of electronic voting rather than voters eschewing electronic devices. In the five locations with electronic voting machines present, approximately 82% of votes cast were done so electronically (41,016 of 50,232 total votes cast). This is a strong indication that, given the choice, ACT voters would choose to use the ACT electronic voting interface rather than paper.

However, the choice of voting interface appears to be correlated with the content of the vote. Table [1](#) shows the percentage share of the vote for each of the three largest parties (Labor, Liberal, Greens), split by region and by voting method.

The table shows, for each region and each party, the percentage share of the paper votes and the percentage share of the electronic votes; the final column shows the factor by which the vote share changes when moving from paper to electronic (that is, it shows the electronic share divided by the paper share).

It is striking that in every region, the two largest parties, Labor and Liberals, gained a higher share of the paper vote than the electronic vote, whereas the Green vote share is much higher in every region in the electronic ballots than in paper.

3.1 Analysis

Perhaps it is not surprising that Green voters are more likely to choose to vote electronically. Familiarity with electronics seems to be negatively correlated with

Table 1. First preferences split by voting interface

Party	Region	Paper %	E-vote %	Factor
Labor	Brindabella	35.88	35.03	0.98
	Ginninderra	41.64	38.12	0.92
	Molonglo	35.06	34.78	0.99
	Overall	37.58	35.77	0.95
Liberal	Brindabella	38.86	36.55	0.94
	Ginninderra	29.32	28.44	0.97
	Molonglo	35.80	33.32	0.93
	Overall	34.22	32.91	0.96
Green	Brindabella	10.24	14.63	1.43
	Ginninderra	11.89	15.37	1.29
	Molonglo	13.66	17.03	1.25
	Overall	12.24	15.89	1.30

age, and Green voters are typically younger. In addition, it is plausible that Green voters may perceive and choose electronic voting as a voting system with a smaller footprint in terms of marginal resource consumption per voter. Furthermore, since the voting machines have special features that encourage usage by voters with visual impairment and voters from a non-English speaking background, there may also be some correlation between these voters and Green voters.

In the ACT, and in Australia as a whole, one might expect the effect of voting technology on the election result to be close to zero. Voting is compulsory in Australia, and one's choice of voting interface should have a negligible affect on the final tally. However, preferential elections suffer from an unusually high rate of invalid voting (roughly 3–4% in most ACT elections, which equates to about 3000 votes per seat), primarily due to voter error in marking ballots, for instance through duplicate or missing rankings [HY07]. For voters who choose to vote electronically, such inadvertent errors are eliminated because the EVACS voting machine notifies voters when they attempt to cast an invalid vote. Thus, in the context of electronic voting it is likely that invalid voting is intentional. Since Green voters have a greater tendency to vote electronically, we would expect lower invalid voting rates among Greens, which would effectively increase the overall Green share of the vote.

We predict that this effect would be much more pronounced if a voting system with only an electronic interface were to be introduced in a jurisdiction without compulsory voting. One's decision on whether or not to bother to vote is likely to be influenced by one's level of comfort with the voting interface. We would expect, therefore, an electronic voting system to encourage turnout among Green voters more than among other voters.

It will be interesting to see whether the same pattern is repeated in the 2012 Legislative Assembly election. If unfamiliarity with electronic systems is a generational phenomenon, the effect will increase as the lower voting age range is filled with technophilic voters and the oldest technophobic voters die; then the effect should stabilize as the first technophiles grow older.

Familiarity with the electronic voting system may also have an effect: if the system was perceived by 2008 voters as easy to use, it might encourage reticent voters to use it in 2012.

4 Ballot Ordering Effects

The organisation of candidate information on ballot papers can influence how voters vote. For example, full-face ballots, which present all the information on a single page, are used almost universally because a ballot paper comprising multiple pages is likely to favour candidates on the first page.

Most jurisdictions worldwide currently require a fixed ballot order for a particular election, consistent across all ballot forms. However, some voting systems, notably Prêt à Voter, mandate a random order of candidates on the ballot paper.

The ACT Legislative Assembly elections are interesting in this regard: the order of the parties is fixed, but the order of candidates within each party is determined by Robson rotation. This makes it possible to analyze the data for ballot ordering effects within parties, by considering whether the votes cast with one order are significantly different from the votes cast with another order; however, because the party order is fixed, it makes it rather harder to glean information as to the effect of the ordering of parties on the ballot paper.

A system like Prêt à Voter is not designed to cope with having several candidates standing for each party, and yet allowing each party's candidates to be grouped together on the ballot paper; in fact, we are not aware of any electronic voting system that allows for this and yet also adheres to the principle of ensuring that the voting terminal does not learn the content of the vote. Enhancing the design of Prêt à Voter to deal with this (not uncommon) situation is worthy of investigation; the results in this section will be informative in assessing the side effects of the various possibilities.

Table 2 shows the vote share achieved by the three main parties when their leaders were, and were not, top of the ballot within their party. The third column shows how many ballots were cast that had a ballot ordering placing the named candidate first within the party, and how many when the named candidate was not listed first; the final column shows the share of the vote received by the party in each case. It will be seen that the effect of the relative ordering of the leader's name has no significant effect on the vote share for that party as a whole. This suggests that ballot order within parties does not significantly affect voters' choice of party.

Table 2. Effect on party vote of name ordering

Candidate	Top?	Ballots	Vote %
Jon Stanhope (Lab)	Yes	11980	40.7
	No	48069	40.0
Zed Seselja (Lib)	Yes	12615	31.2
	No	75651	31.6
Meredith Hunter (Green)	Yes	29955	14.1
	No	30094	13.7

4.1 Donkey Votes

One natural consequence of compulsory voting is that many people who have no interest in politics or in who wins the election are forced to vote. Some will cast blank or spoiled ballots ('informal votes'); but there is evidence that quite a number of people cast votes that give preferences according to the printed order on the ballot paper, so as to fulfil their legal duties with minimum effort. These votes are termed 'donkey votes'.

A previous study [TECOS] considered some forms of donkey voting in the ACT 1998 election, and concludes that 22% of votes were donkey votes. However, this study considers only voting behaviour that is indifferent to the ordering of candidates within a party, and does not look at votes that are indifferent to the relative ordering of the parties themselves. A 'linear vote', in the terminology of [TECOS], is one that, for each party, gives its rankings in ascending order down the ballot paper, and so is sensitive to the printed order of the candidates within each party. A 'circular vote' is one that selects a particular candidate within each party, and numbers downwards from that candidate, and then continues the ranking from the top. Both linear votes and circular votes do demonstrate some choice on the part of the voter, and are therefore not full donkey votes.

The ACT uses Robson rotation to counter the effect of linear and circular voting by distributing such votes evenly across all candidates within a party. However, the ordering of parties on the ballot paper is drawn by lots before the election, and, for any given region, is the same for every ballot paper.

It is interesting to ask how many donkey votes there were in the ACT 2008 election. We will consider here only votes that show no clear preference for one party over another.

In addition, in this section we will consider only votes that expressed a full ranking. The reason for this is that these are the clearest candidates for donkey votes. The quickest way to cast a legitimate ballot is to write a '1' in the top right box and then stop¹; but since this is a plausible genuine expression of preference, it is impossible to determine which of these votes are donkey votes.

¹ Despite the wording on the ballot papers, votes that choose only one candidate are treated as formal votes.

Similar reasoning applies to votes that fill in the first column and then stop. If the party order were randomized on the ballot papers, it would be possible to analyze the effect of the ordering; but unfortunately the ordering is fixed for each region.

Row-Monotonic Votes. A vote is *row-monotonic* if the projection that maps each ranking to the row in which it appears is monotonic; in other words, the vote is filled in from top to bottom down the ballot paper, but without any constraint on the relative rankings within a row.

There were very few row-monotonic votes in the ACT 2008 election: 4 in Brindabella (of 63,334 ballots in total), 3 in Ginninderra (of 60,049) and 3 in Molonglo (of 88,266). In each region, one of these is also monotonic within each row; that is, the vote is filled in from left to right (or vice versa) across the top row, and then the same across the second row, and so on. (Not all columns are the same length, so not every row has the same number of candidates in it.)

It seems that row-monotonic donkey votes are not a significant problem. This is perhaps because the boxes in each column are nearer together than the boxes within each row, and so it is less effort to fill in the columns one by one. This is where we turn next.

Column-Monotonic Votes. A vote is *column-monotonic* if the projection that maps each ranking to the column in which it appears is monotonic; in other words, the vote is filled in from left to right or right to left across the ballot paper, but without any constraint on the relative rankings within a column.

In Brindabella, there were 828 column-monotonic votes. Of these, 521 went left to right, and 307 went right to left. These seem natural candidates for donkey votes; the 120 ballots that were fully monotonic (the ranking goes down the first column, then down the second, then down the third, and so on) in particular show strong evidence in this direction.

However, these numbers are in stark contrast to those from Ginninderra and Molonglo. In Ginninderra, there were 16 column-monotonic votes, all of which went left to right; in Molonglo there were 19 that went left to right and 2 that went right to left. Ginninderra and Molonglo each had only 9 fully monotonic ballots.

This gives Brindabella a column-monotonic rate of 1.3%, compared with 0.026% for Ginninderra and 0.021% for Molonglo. Why so much higher in Brindabella?

The answer appears to be related to the ordering of the parties on the ballot papers in the three regions. Table 3 below shows the party ordering for each region. The Brindabella party ordering is, purely as a result of the random draw, an ordering that may well fit in with many voters' preferences, either in its listed direction or in reverse. This is all the more so because the Liberal party and the Labor party make a point of instructing their followers (not just in the ACT but nationwide) to put the other last in their ranking, and generally adopt this policy when constructing how-to-vote cards. Such cards are not used in the ACT; but the thinking may well be adopted by ACT voters in any case. The

Table 3. Party order on ballots in each region

Brindabella	Ginninderra	Molonglo
Liberal	Motorists	Pangallo Independents
Community Alliance	Labor	Labor
Motorists	Community Alliance	Community Alliance
Greens	Greens	Richard Mulcahy
Labor	Liberal	Motorists
	Ungrouped	Liberal Democrats
	Ungrouped	Greens
		Liberals
		Ungrouped

third largest party, the Green Party, is usually more friendly to the Labor party than to the Liberals, and is in coalition with the Labor party in both the Federal House of Representatives and the Tasmanian Parliament. A Liberal voter who distrusts the Greens and Labor is reasonably likely to vote in the listed order, and a Labor voter who distrusts the Liberals but is amenable to the Greens is reasonably likely to vote in reverse order.

The Ginninderra and Molonglo orderings, however, are much less natural orderings of the parties. In particular, none of the three main parties is listed first or last. A carefully considered column-monotonic vote in Ginninderra or Molonglo would require an idiosyncratic political persuasion, to say the least.

It seems likely, then, that most of the ballots in Brindabella that appear to be full donkey votes are in fact sincerely cast ballots that accord roughly with the voters' preferences. (It is possible that the preference ordering of the two very small parties, the Community Alliance and the Motorists, is still affected by ballot ordering.) This makes for an instructive cautionary tale: if we had only the Brindabella data, we might be all too ready to class all of the fully monotonic ballots as donkey votes.

Moral: think carefully before you call someone a donkey.

5 Expressivity

ACT Legislative Assembly elections allow voters to express as many preferences as they wish, from choosing a single candidate to giving a full ranking of all candidates. Since STV satisfies the later-no-harm criterion, there is no rational basis for not submitting a full ranking, except if the time cost of filling in the ballot is considered to outweigh the potential benefits of changing the result.

One might therefore hope that a well designed electronic system would encourage voters to be more expressive, and to submit a fuller ranking than they would otherwise do. Indeed, a first pass of the 2008 data suggests that there is a small correlation between voting interface and expressivity.

Table 4. Number of preferences expressed by paper and electronic voters

Region	Seats/Cands	Interface	Votes	Expressivity
Brindabella	5/19	Paper	2149	7.07
		Electronic	11740	7.37
Ginninderra	5/27	Paper	3254	7.18
		Electronic	11244	7.59
Molonglo	7/40	Paper	3813	9.67
		Electronic	18032	9.75

Table 4 shows a measure of expressivity for paper and electronic voters in the five regions where both interfaces were offered. The final column shows the mean average number of candidates ranked by the voters. The figures are quite compelling: in each region, the number of preferences given by electronic voters was higher than that given by the paper voters.

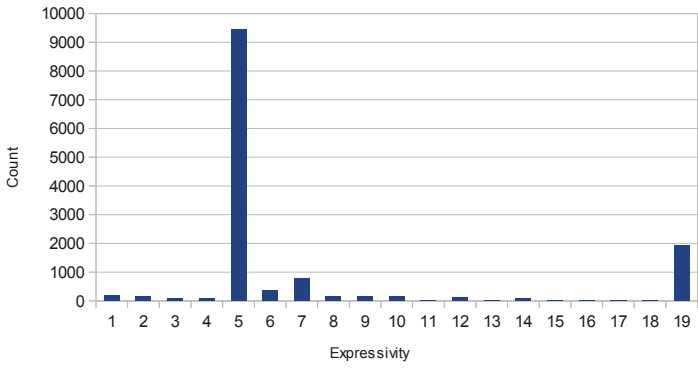
There were more candidates in Ginninderra than in Brindabella, which has a small effect on expressivity; but it seems that the number of seats is the key indicator. This is because the majority of voters choose a party, rank the candidates from their chosen party, and then stop; and since the largest parties field the same number of candidates as there are seats, most voters' rankings are the same length as the number of seats.

Alas, this behaviour is what makes the numbers appear more useful than they in fact are. The extra expressivity of electronic voters turns out to be a side effect of the correlation with party choice we have already noted in Section 3. Figure 2 shows, for each possible length of ranking, how many voters cast a ranking of that length. The charts are broken down according to region.

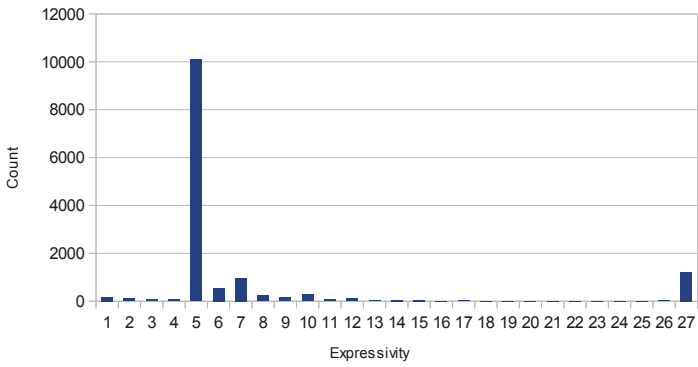
The huge numbers expressing the same number of preferences as there are seats is immediately apparent: the peak dominates the skyline in each case. The second highest peak is consistently for those who diligently give a complete ranking of all candidates. (It is perhaps slightly disappointing from a mathematical perspective that so few voters think to save time and energy by eliding the final preference!)

What next catches the eye are the local maxima evident at 7 (Brindabella and Ginninderra) and 10 (Molonglo). How do we explain these?

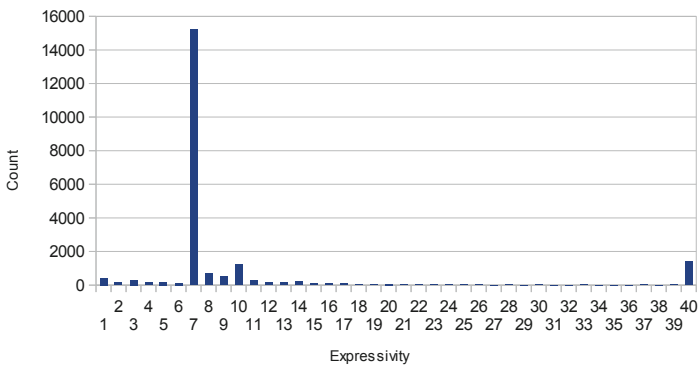
Australian politics is dominated by the Labor Party and the Liberal Party. In the previous (2004) election, the Green party had won one seat in Molonglo, and the rest of the seats were shared by Labor (nine) and the Liberals (seven). For this reason, most other parties do not consider it worthwhile to field as many candidates as there are seats. (The only other party to field a full set of candidates, even in one region, was the Australian Motorist Party, with around 5% of first preferences across the ACT.) The Green party thus chose to field two candidates in Brindabella, two in Ginninderra and three in Molonglo.



(a) Brindabella



(b) Ginninderra



(c) Molonglo

Fig. 2. Expressivity in the ACT

This explains the local maxima. In each region, the peak comes at the sum (7/7/10) of the number of seats (5/5/7) and the number of Green party candidates (2/2/3). Many of the voters who favour one of the two dominant parties rank the candidates from their party and then stop; a large proportion of Green voters, on the other hand, know that their candidates are less likely to win than those under the Labor or Liberal banner, and so, having ranked the Green candidates in order, they then choose either Labor or Liberal and rank those candidates too.

So Labor and Liberal voters, on the whole, tended to cast rankings of length 5, 5 and 7 in Brindabella, Ginninderra and Molonglo respectively; many Green voters cast rankings of length 7, 7 and 10. This gives us the local maxima at 7/7/10; but it also explains the greater expressivity in the electronically cast ballots. Voting Green is positively correlated with voting electronically; and voting Green is also positively correlated with greater expressivity.

The ACT data, therefore, does not support the conclusion that electronic voting encourages voters to give a fuller ranking of candidates.

However there are some indications elsewhere in Australia that electronic voting might increase expressiveness. In the 2007 Federal Election, the Australian Electoral Commission trialled two electronic voting systems. One trial used a remote voting system for Australian Defence Force (ADF) personnel deployed overseas and the other used special voting machines for blind and visually impaired (BVI) voters.

Ballot papers in Federal STV elections enable voters to select a group voting ticket rather than marking their own preferences, with the latter commonly referred to as voting 'below-the-line'. Federal elections are very strict in that below-the-line voting requires voters to rank every single candidate (and in many cases there can be around 80 candidates). As a result of these onerous rules, only 3.2% of voters overall voted below-the-line in 2007, which is quite typical at Federal level. But post-election evaluations of the trial systems revealed that voters who used the electronic voting systems were more likely to vote this way: 5.2% in the ADF trial [AEC08b] and 10% in the BVI trial [AEC08a]. The ADF evaluation found a correlation between network speeds and voting below-the-line, and this suggests that voters make a conscious trade-off between the desire to be expressive and the time it takes to cast a vote.

Although no firm conclusions can be drawn from a data sample of this size, it is remarkable that such a high proportion of blind and visually impaired voters who participated in the trial chose to mark all the preferences in spite of the obvious burden in doing so. A possible reason is that BVI voters were making the most of the opportunity to cast a secret ballot: the BVI evaluation found high levels of satisfaction with voter privacy.

6 'Italian' Attacks

Many secure electronic voting systems require the full ballot data to be published after the election, to enable auditing. Prêt à Voter, for instance, relies for its

integrity on publication of both encrypted and unencrypted sets of ballots, with a proof that the sets correspond to the same ballots, but without any information as to the mapping between the sets. Some work has been done on tallying STV elections without disclosing the full ballot data [Hea07, BMN⁺09], but these solutions are expensive, and if full disclosure can be justified then it is a more efficient option.

One potential weakness of full disclosure is that it allows so-called ‘Italian’ attacks [Hea07, DC07], where a coercer can bribe and intimidate voters to vote for prescribed candidates. As a simple example, each coerced voter can be instructed to choose a specified candidate as the first preference, then mark the lower preferences so that the ordering forms a uniquely identifiable ‘signature’. After the election the coercer uses the electronic ballot data to check which signatures appear, and hence which voters complied with their given instructions. In this way the anonymity of the secret ballot is violated.

How easy is it to mount an Italian attack that will not be obvious from the published ballots? Will such coerced ballots stand out as being atypical voter behaviour?

This is a difficult question to answer definitively, because it depends on what qualifies as ‘typical’ voter behaviour: what is needed is a coherent and accurate model of how voters fill in their ballots. It may be possible to construct such a model from the ACT ballot data, and this is something that we plan to consider in future work; however, there are some natural options for an Italian attacker that appear attractive with the analysis that we have conducted so far.

The easiest strategy for such an attacker is to specify a sequence of candidates who must be placed at the start of the ranking, and then some random sequence of candidates to follow. This random sequence must be different for each voter whom the attacker wishes to coerce, so that he can determine individual compliance. One might hope that such ballots would be readily identifiable: voters do not cast their ballots randomly, but tend to give similar rankings to similar candidates, and in particular to candidates from the same party.

The ACT ballots do not suggest that these attacks would be easy to spot: there are, in fact, quite a number of voters who vote seemingly randomly. One measure of this is the number of *party switches* on a ballot—that is, the number of instances of two consecutive preferences that do not select candidates from the same party. And indeed, on this measure, most voters do fill in ballots with a small number of party switches. But a surprising number of voters switch parties at almost every turn.

In Molonglo, for instance, there were a total of 88,266 ballots cast. Most of these were for candidates from a single party, and thus contained no switches. A total of 53,867 ballots contained between one and ten switches. But 63 ballots contained more than thirty party switches; and 7 ballots contained thirty-nine switches. Since there were only forty candidates, this means that seven voters took the trouble to construct a ranking that never selected the same party for two consecutive candidates! Similar observations apply to Brindabella and Ginninderra. Some of these are row-monotonic donkey votes discussed in

Section 4.1, but this by no means accounts for all of the ballots with a high number of switches.

There are only two possibilities. Either Italian attacks are already taking place and are present in the ACT data—a fascinating point for further investigation, but it seems unlikely—or such attacks have plenty of room for hiding. An attacker who wants scope for coercing 40,000 voters (quite an operation) could keep all the ballots having at most eight party switches even if he always chose the same eight candidates from which to choose random orderings and they all came from different parties.

It seems likely, then, that without substantially more sophisticated models of voter behaviour, evidence of Italian attacks will be hard to detect from the published ballot data.

7 Conclusion and Future Work

In this paper, we have drawn various lessons from the ACT 2008 ballot data. These lessons are vital considerations when discussing the possibility of changing the electoral system, because it is important to think through not just the primary effects of such changes but also the secondary, often unintentional, effects.

Our ultimate aim is to use ballot data to construct a model of voter behaviour. However, it may well be the case that real world election data is not sufficient to model voter behaviour. Although the ACT data is rich, it does not contain information such as which voters chose a language other than English, or used features for the visually impaired; nor does it log the time at which each ballot was cast. Such fine-grained details would help to rule out certain factors when investigating trends and specific hypotheses, though they are also likely to violate anonymity, and so it seems unlikely that this level of detail will ever be forthcoming for governmental elections. Further studies are needed to determine how to obtain more definitive conclusions.

There appears to be growing interest in electoral systems that allow greater expressivity than that afforded by first-past-the-post: for example, there have recently been (unsuccessful) referenda for the alternative vote in the UK, and STV in Canada. If these systems were to be adopted, then e-voting might have an increased impact on voter behaviour and the election outcome. Possible positive influences might include reductions in invalid voting and donkey voting, while negative influences might include voter confusion due to poor user interface design, which could for instance cause voters to erroneously cast valid votes that are contrary to their intention. The potential for such consequences warrants careful assessment when considering the adoption of e-voting and/or new electoral systems.

More generally, electronic election systems may have already had an impact on increasing expressivity. It would seem that large-scale STV elections are feasible only with e-counting because the counting procedure is too complex to perform manually. Without e-counting, it is improbable that STV would have been used for Iceland's 2010 Constitutional Assembly, which elected 25 delegates

from 525 candidates. In Australia, it appears that STV would not have become so entrenched without e-counting. For example, New South Wales Legislative Council elections use a highly complicated version of STV and have 21 seats, over 300 candidates and 4.5 million voters. Counting the votes manually would simply not be possible.

References

- [ACT08] Australian Capital Territory Electoral Commission: Ballot paper preference data – 2008 Election (2008)
- [AEC08a] Australian Electoral Commission: Final Evaluation Report: Evaluation of the Electronic Voting Trial for Blind and Sight Impaired Electors at the 2007 Federal Election (2008)
- [AEC08b] Australian Electoral Commission: Final Evaluation Report: Evaluation of the Remote Electronic Voting Trial for Overseas Based ADF Personnel Electors at the 2007 Federal Election (2008)
- [AHT09] Alvarez, M.R., Hall, T.E., Trechsel, A.H.: Internet Voting in Comparative Perspective: The Case of Estonia. *Political Science Politics* 42(03), 497–505 (2009)
- [BMN⁺09] Benaloh, J., Moran, T., Naish, L., Ramchen, K., Teague, V.: Shuffle-Sum: Coercion-Resistant Verifiable Tallying for STV Voting. *IEEE Transactions on Information Forensics and Security* (2009)
- [DC07] Di Cosmo, R.: On Privacy and Anonymity in Electronic and Non Electronic Voting: the Ballot-As-Signature Attack. HAL Open Archive Document hal-00142440, version 2 (2007)
- [Hea07] Heather, J.A.: Implementing STV Securely in Prêt à Voter. In: *Proceedings of the 20th IEEE Computer Security Foundations Symposium, Venice, Italy*, pp. 157–169 (July 2007)
- [HMP09] Hall, T.E., Quin Monson, J., Patterson, K.D.: The human dimension of elections: How poll workers shape public confidence in elections. *Political Research Quarterly* 62(3), 507–522 (2009)
- [HPT⁺10] Hanmer, M.J., Park, W.-H., Traugott, M.W., Niemi, R.G., Herrnson, P.S., Bederson, B.B., Conrad, F.C.: Losing fewer votes. *Political Research Quarterly* 63(1), 129–142 (2010)
- [HY07] Hill, L., Young, S.: Protest or error? Informal voting and compulsory voting. *Australian Journal of Political Science* 42(3), 515–521 (2007)
- [RBH⁺09] Ryan, P.Y.A., Bismark, D., Heather, J.A., Schneider, S.A., Xia, Z.: The Prêt à Voter Verifiable Election System. *IEEE Transactions on Information Forensics and Security* 4(4), 662–673 (2009)
- [TEC08] Tasmanian Electoral Commission: A Discussion Paper on Robson Rotation in Tasmania (2008)

Author Index

- Allepuz, Jordi Puiggali 36
- Buckland, Richard 224
- Bursuc, Sergiu 190
- Castelló, Sandra Guasch 36
- Chaum, David 140
- Culnane, Chris 174
- Demirel, Denise 158
- DeYoung, Henry 53
- Essex, Aleksander 122
- Florescu, Alex 140
- Frosch, Tilman 89
- Gjøsteen, Kristian 1
- Grewal, Gurchetan S. 190
- Haenni, Rolf 71
- Heather, James 174, 242
- Heiberg, Sven 208
- Heiderich, Mario 89
- Hengartner, Urs 122
- Henning, Maria 158
- Henrich, Christian 122
- Joaquim, Rui 104
- Koenig, Reto 19, 71
- Laud, Peeter 208
- Llewellyn, Morgan 242
- Nandi, Mridul 140
- Niemietz, Marcus 89
- Popoveniuc, Stefan 140
- Ribeiro, Carlos 104
- Rubio, Jan 140
- Ryan, Mark D. 190
- Ryan, Peter Y.A. 158
- Schläpfer, Michael 71
- Schneider, Steve 158, 174
- Schürmann, Carsten 53
- Schwenk, Jörg 89
- Spycher, Oliver 19, 71
- Srinivasan, Sriramkrishnan 174
- Teague, Vanessa 224, 242
- Volkamer, Melanie 19, 158
- Vora, Poorvi L. 140
- Wen, Roland 224, 242
- Willemson, Jan 208
- Xia, Zhe 174
- Zagórski, Filip 140