Analysis of an Electronic Boardroom Voting System[*]

Mathilde Arnaud, Véronique Cortier, and Cyrille Wiedling

LORIA - CNRS, Nancy, France

Abstract. We study a simple electronic boardroom voting system. While most existing systems rely on opaque electronic devices, a scientific committee of a research institute (the CNRS Section 07) has recently proposed an alternative system. Despite its simplicity (in particular, no use of cryptography), each voter can check that the outcome of the election corresponds to the votes, without having to trust the devices.

In this paper, we present three versions of this system, exhibiting potential attacks. We then formally model the system in the applied pi-calculus, and prove that two versions ensure both vote correctness (even if the devices are corrupted) and ballot secrecy (assuming the devices are honest).

Keywords: Ballot Secrecy, Boardroom Voting, Correctness, Formal Methods.

1 Introduction

Electronic voting has garnered a lot of attention in the past years. Most of the results in this field have been focused on two main types of settings: distant electronic voting and voting machines. Distant electronic voting corresponds to systems where voters can vote from their own computers, provided they are connected to the Internet. Many systems have been devised, including academic ones (e.g. Helios [2], Civitas [5], or FOO [10]). Voting machines are used in polling stations and speed up the tally. Examples of voting machines are e.g. the Diebold machines [9] or the Indian voting machines [19], both of them having been subject to attacks [9,19].

Several security notions have been proposed for voting systems and can be split into two main categories: privacy [8] and verifiability [14]. Privacy ranges from ballot secrecy to coercion-resistance and ensures that no one can know how a particular voter voted. Verifiability enables voters to audit the voting process, e.g. by checking that their ballots appear on the bulletin board (individual verifiability), or checking that the outcome of the election corresponds to the ballots on the bulletin board (universal verifiability).

In this paper, we focus on a different and particular setting: boardroom meetings. Many committee meetings require their members to vote on several motions/decisions. Three techniques are typically used.

- Show of hands: this is a simple and cheap technique, which offers no privacy and requires to count the raised hands.

[*] The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement no 258865, project ProSecure.

J. Heather, S. Schneider, and V. Teague (Eds.): VoteID 2013, LNCS 7985, pp. 109–126, 2013.
© Springer-Verlag Berlin Heidelberg 2013

- Paper ballot: this solution offers privacy but may be tedious, in particular when there are several rounds of vote during a meeting.
- Use of electronic devices.

Electronic devices seem to offer both simplicity of use and privacy: committee members simply need to (privately) push a button corresponding to their choice on their own device and a central device computes and publishes the result. However, these systems are opaque: what if someone controls the central device and therefore falsifies the result of the election? In many committees such as boarding committees or scientific councils, controlling the result of the election (e.g. choice of a new president, decision on the future of a company, *etc.*) is even more important in terms of impact than breaking privacy. Even if the system is not malicious, it can simply dysfunction with no notifications, as witnessed e.g. by the "CNRS Section 07" committee members (the scientific council in Computer Science of the CNRS, a French national research institute). In response to these dysfunctions, a subgroup of the CNRS Section 07 committee members, namely Bruno Durand, Chantal Enguehard, Marc-Olivier Killijian and Philippe Schnoebelen, with the help of Stefan Merz and Blaise Genest, have proposed a new voting system that is meant to achieve:

- simplicity: it could be easily adapted to existing devices
- privacy
- full verifiability, even if the electronic devices are corrupted

A few other systems tailored to boardroom election have been proposed such as [11,12]. A feature of the "CNRS Section 07" system is that it does not use cryptography, which makes the system easier to understand and trust, for non experts.

Our contributions. We provide a full review of the voting system proposed by the CNRS Section 07, illustrating the applicability of formal models and in particular, the applicability of the latest definitions and the proof techniques in formal methods. The key idea of the CNRS Section 07 voting system is that each vote appears on the screen, together with a unique identifier (randomly generated by the central device). This unique identifier allows voters to check that their votes have been counted. Due to our attacks on the initial version (that called $\mathsf{F2FV}^1$), two variants of it have been proposed: in $\mathsf{F2FV}^2$, the random identifier is generated by both the ballot box and the voter while in $\mathsf{F2FV}^3$, the random identifier is generated by the voter only. It is interesting to note that this last version is actually close to the protocol devised by Bruce Schneier in [18].

We first describe the three versions and we review in details the possible attacks:

- The initial version $\mathsf{F2FV}^1$ is subject to a “clash-attack”, using the terminology of [16]. The attack works roughly as follows: if the same identifier is used for two different voters that voted the same way, then a dishonest ballot box may replace one of the ballots by any ballot of its choice. The last version $\mathsf{F2FV}^3$ (and thus the Schneier’s protocol as well) suffers from the same attack (with relatively small probability) if the random numbers are small, which is likely to be the case in practice.

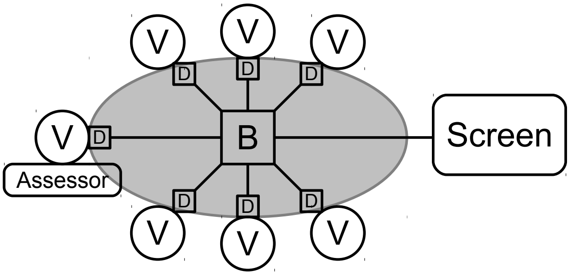


Fig. 1. Schema of the election

- The other attacks are against privacy. Obviously, a dishonest ballot box may know how any voter voted. We discuss other ways for a dishonest ballot box to break privacy. One of the attack works even if the ballot box does not initially know to which a ballot belongs to.

To conduct a more thorough security analysis, we formally model these systems in the applied pi-calculus [1], a process algebra well adapted to security protocols. Computational models where attackers are modeled by polynomial time probabilistic Turing machines are, as a rule, more accurate. However, since the systems here involve no cryptography, we chose the simplicity of the applied pi-calculus, for which several security analyses of voting protocols have already been conducted (e.g. [6,7]).

We focus on two main security properties: vote correctness and privacy. The CNRS Section 07 voting system is primarily designed to ensure that, even if all the electronic devices are corrupted, any approved election outcome reflects the votes of all voters. This property has been introduced by Benaloh and Tuinstra [3] and more precisely defined by Catalano *et al* in [13] and is called *correctness*. We provide a formal definition of this property and prove that the two versions $\mathsf{F2FV}^2$ and $\mathsf{F2FV}^3$ ensure vote correctness, even if all devices are corrupted (but assuming voters use random numbers). In contrast, privacy cannot be ensured when the central device is corrupted. However, privacy is guaranteed against external users (including voters). Formally, we show privacy for the well established notion of privacy defined in [8], assuming that the electronic devices are honest.

2 Setting

We consider a particular setting, typically for boardroom meetings, where all voters are present in the same room and are given a dedicated voting equipment. In what follows, we assume the individual devices to be linked to a central device. The central device is responsible for collecting the ballots and publishing them. Such systems are standard in many committees (e.g. parliamentary assembly, corporate boards, *etc.*). The particularity of the voting system (and its variants) proposed by the CNRS Section 07 is that it assumes the presence of a screen that each voter can see. This screen ensures that all voters simultaneously see the same data and is the key element for the voting system.

Specifically, the system involves voters and their electronic voting devices, a ballot box (the central device), and a screen. Moreover, a voter is chosen to take on the role of an assessor (for example the president of the committee or her secretary). This is illustrated in Figure 1.

Ballot box. The ballot box is the central device that collects the ballots and tallies the votes. It communicates with the electronic devices of the voters over private individual channels. Once the voting phase is over, the ballot box publishes the outcome of the election on the screen.

Screen. The screen displays the outcome of the election for validation by the voters and the assessor. Since the voters are in the same room, they all see the same screen.

Voter. The voter role has two phases. In the first phase, he casts his vote through her electronic device. In the second phase, he performs some consistency checks looking at the screen and lets the assessor know whether his checks were successful, in which case he approves the procedure.

Personal voting device. Each individual voting device has a pad or some buttons for the voter to express her choice. The device communicates the value of the vote entered by the voter directly to the ballot box.

Assessor. The assessor is a role that can be performed by any voter. He does not hold any secret. He is chosen before the execution of the protocol. The assessor is responsible of some additional verifications. In particular, he checks that each voter has approved the procedure. If one voter has not, he must cancel the vote and start a new one.

3 Face-to-Face Voting System

We describe in details the electronic boardroom voting system designed by the CNRS Section 07 committee. We actually present three versions of it. The three versions have in common the fact that the central device and/or the voters generate a random number that is attached to the vote. Both the vote and the random number are displayed on the screen. This way, each voter can check that his vote (uniquely identified by its random number) is counted in the tally. We could have presented the version that offers the best security guarantees but we think the flaws in the other versions are of interest as well. The three versions differ in who generates the randomness:

- Initial version: The ballot box generates the random identifier for each voter.
- Second version: Both the ballot box and voters generate a random identifier.
- Third version: The voters generate their identifiers.

The three voting systems are summarized in Figure 2 and are described in details in the rest of the section. Since the votes are transmitted in clear to the central device on uniquely identified wires, ballot secrecy is clearly not guaranteed as soon as the central device is corrupted. So for ballot secrecy, we assume that the central device behaves honestly, that is, the secrecy of the ballots will be guaranteed only against external users (including the voters themselves). The major interest of the CNRS Section 07 system is that it ensures vote correctness *even if the central device is corrupted*, that is the voters do not need to trust any part of the infrastructure.

Note that in practice, the "random numbers" used in the remaining of the paper should typically be numbers of 3-4 digits, so that they are easy to copy and compare.

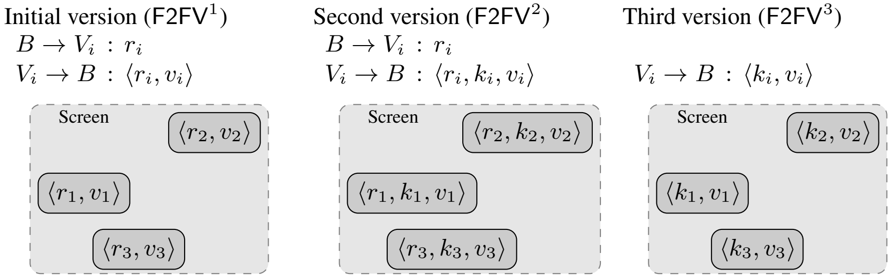

Initial version (F2FV^1)
$B \to V_i : r_i$
$V_i \to B : \langle r_i, v_i \rangle$
Screen
$\langle r_2, v_2 \rangle$
$\langle r_1, v_1 \rangle$
$\langle r_3, v_3 \rangle$
Second version (F2FV^2)
$B \to V_i : r_i$
$V_i \to B : \langle r_i, k_i, v_i \rangle$
Screen
$\langle r_2, k_2, v_2 \rangle$
$\langle r_1, k_1, v_1 \rangle$
$\langle r_3, k_3, v_3 \rangle$
Third version (F2FV^3)
$V_i \to B : \langle k_i, v_i \rangle$
Screen
$\langle k_2, v_2 \rangle$
$\langle k_1, v_1 \rangle$
$\langle k_3, v_3 \rangle$

Fig. 2. Voting processes

3.1 Initial System F2FV^1

Voting Phase. The ballot box B starts the election by generating a random number r for each voter V, and sends this random number to the voter. The voter V receives the random number r, uses it to form his ballot $\langle r, v \rangle$ where v is his vote, and sends his ballot to the ballot box. Finally, all the ballots $\langle r, v \rangle$ are displayed on the screen E. This marks the end of the voting process.

Validation Phase. The validation part can then begin. Each voter checks that his ballot is correctly included in the list of ballots displayed on the screen. The assessor waits for each voter to state that his vote appears on the screen. He also checks that the number of ballots matches the number of voters. If all checks succeed, the assessor approves the outcome of the election.

Possible Attacks. The key idea of this system is that each random identifier should be unique, ensuring a one-to-one correspondence between the votes that appear on the screen and the votes cast by the voters. However, a corrupted ballot box may still insert ballots of its choice, mounting a so-called "clash-attack" [16]. The attack works as follows: the (dishonest) ballot box guesses that two voters Alice and Bob are going to vote in the same way. (This could be a pure guess or based on statistical analysis of the previous votes.) The ballot box then sends the *same* nonce r to Alice and Bob. Since Alice and Bob cast the same vote v, they both send back the same ballot $\langle r, v \rangle$. The ballot box is then free to display $\langle r, v \rangle$ only once and then add any ballot of its choice. Both Alice and Bob would recognize $\langle r, v \rangle$ as their own ballot so the result would be validated.

For example, assume there are three voters A, B, and C and the ballot box guesses that A and B vote identically. Suppose A and B cast 0 and C casts 1. The ballot box can replace the two votes for 0 by one vote for 0 and one vote for 1, making the "1" vote win. This can be done by simply sending the same randomness r_a to both A and B.

$$\begin{array}{lll} B(I) \to V_A : r_a & B(I) \to V_B : r_a & B(I) \to V_C : r_c \\ V_A \to B(I) : \langle r_a, 0 \rangle & V_B \to B(I) : \langle r_a, 0 \rangle & V_C \to B(I) : \langle r_c, 1 \rangle \\ & B(I) \to E : \langle r_a, 0 \rangle & \\ & B(I) \to E : \langle r_b, 1 \rangle & \\ & B(I) \to E : \langle r_c, 1 \rangle & \end{array}$$

3.2 Second System $\mathsf{F2FV}^2$

The attack on the initial system $\mathsf{F2FV}^1$ is due to the fact that the ballot box may cheat when generating random unique identifiers. So a second solution has been proposed, where both the voters and the ballot box generate a part of the random identifier.

Voting Phase. The ballot box B starts the election by generating a random number r for each voter V, then sends this random number to the voter. The voter V receives the random number r, picks a new random number k (possibly using a pre-generated list), and uses it to form his ballot $\langle r, k, v\rangle$ where v is his vote, and then sends his ballot to the ballot box. Finally, all the ballots $\langle r, k, v\rangle$ are displayed on the screen E.

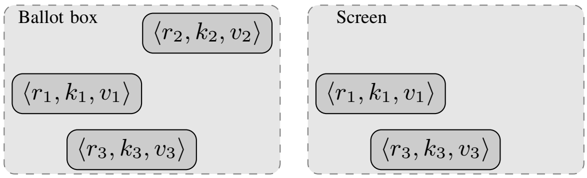
The validation phase works like for the protocol $\mathsf{F2FV}^1$.

Possible Attacks. As we shall see in Section 5.2, this second version ensures vote correctness, even if the ballot box is corrupted. As for the two other variants, privacy is not guaranteed as soon as the central device (the ballot box) is corrupted. Indeed, the central device may leak how each voter has voted or may record it on some memory. However, such attacks against privacy assume a rather strong control of the ballot box, where the attacker can access to the device either during or after the election. We further discuss some more subtle flaws which require a lower level of corruption We describe two different attacks.

Encoding information in the randoms. As already mentioned, a fully corrupted ballot box may transmit how each voter voted since it receives the votes in the clear, from uniquely identified wires. However, $\mathsf{F2FV}^2$ (and $\mathsf{F2FV}^1$) also suffers from offline attacks, where an attacker simply logs the election outcome. Indeed, it makes sense anyway to keep a copy of the screen after each election. The attack works as follows. Instead of generating fully random numbers, the ballot box could be programmed to provide a voter i (where i is the number identifying the voting device used by the voter) with a nonce r_i such that $r_i \equiv i \mod p$, where p is larger than the number of voters. In this way, an intruder could deduce from a ballot $\langle r, k, v\rangle$ the identity of the voter, simply by computing r modulo p. Of course, the identity of the voters could be encoded in the randomness in many other ways, making the detection of such an attack very unlikely. This attack simply assumes the attacker had access to the central device, at least once prior to the election (e.g. during its manufacturing). It does not require the attacker to access the ballot box during nor after the election.

Swallowing ballots. There is a more direct (but easily detectable) way to break privacy, as sketched in Figure 3. Indeed, assume an attacker wants to know to whom a ballot $\langle r_2, k_2, v_2\rangle$ belongs to. In case the attacker simply controls the display of the screen, he can send a modified set of ballots to the screen. E.g. if he sends $\langle r_2, k_2, v_2'\rangle$ instead of $\langle r_2, k_2, v_2\rangle$), or if he simply remove this ballot, the voter who submitted the ballot $\langle r_2, k_2, v_2\rangle$ would then complain, revealing his identity.

Security Guarantees. We show in Section 5 that this second version ensures vote correctness, even if the ballot box is corrupted. It also ensures ballot secrecy, assuming the ballot box is honest.


Ballot box
$\langle r_2, k_2, v_2 \rangle$
$\langle r_1, k_1, v_1 \rangle$
$\langle r_3, k_3, v_3 \rangle$
Screen
$\langle r_1, k_1, v_1 \rangle$
$\langle r_3, k_3, v_3 \rangle$

The ballot $\langle r_2, k_2, v_2 \rangle$ is not sent on the screen. Voter V_2 reports his ballot is missing, leaking how he voted to the attacker.

Fig. 3. Attack against ballot secrecy

3.3 Third System F2FV^3

To circumvent the privacy issue of the second system, when the ballot box is somewhat honest (the attacker cannot access not interfere with it) but has been maliciously programmed, a third version has been proposed, where the random identifier is generated by the voter only.

Voting Phase. Each voter V picks a random number k and uses it to form his ballot $\langle k, v \rangle$ where v is his vote, and then sends his ballot to the ballot box. All the ballots $\langle k, v \rangle$ are displayed on the screen E.

The validation phase works like for systems F2FV^1 and F2FV^2.

Possible Attack. This third system is vulnerable to the same kind of attacks against vote correctness as the one described for system F2FV^1. Indeed, in case two voters pick the same random number and vote for the same candidate, for instance $(k_A, v_A) = (k_B, v_B)$, the ballot box could remove one of these ballots and replace it by a ballot of its choice without being detected. Note that, due to the birthday theorem, it is not so unlikely that two voters use the same random number. For example, assume voters use 4 digits numbers. Then there is a probability of more than 0.2 to have a collision in a room of 67 members and more than 0.5 in a room of 118 members. In case, only 3 digits numbers are used, there is already a probability of collision of about 0.5 for only 37 members. These figures assume that the voters pick true random numbers. In case they generate numbers "manually", the entropy is usually much lower (e.g. users are sometimes reluctant to generate numbers with repeated digits). In such cases, the probability of collision increases accordingly.

As mentioned in the introduction, the voting protocol proposed by Bruce Schneier in [18] being very similar, it suffers from the same attack.

Security Guarantees. We show in Section 5 that this third version ensures vote correctness, even if the ballot box is corrupted (providing voters generate true randomness). It also ensures ballot secrecy, assuming the ballot box is honest.

3.4 Common Weaknesses

If a voter claims that her ballot does not appear on the screen, then the election round is canceled and everyone has to vote again. This means that a dishonest voter may choose to cancel an election (e.g. if she's not happy with the result), simply by wrongly claiming that her vote does not appear. This is mitigated by the fact that the advantage of the attack is small (the election just takes place again) and the voter could be blamed as being dishonest or inattentive if this happens too often.

4 Formal Model

The remaining of the paper is devoted to the formal proof of security of ballot privacy and vote correctness for the two systems $\mathsf{F2FV}^2$ and $\mathsf{F2FV}^3$. We use the applied pi-calculus [1] for the formal description of the voting systems. We briefly recall here all the definitions of the applied pi-calculus.

4.1 Syntax

Messages are represented by *terms* built on an infinite set $\mathcal{N}$ of *names* (used to name communication channels or atomic data), a set $\mathcal{X}$ of *variables* and a *signature* Σ, which is a finite set of *function symbols* representing primitives. Since our voting systems do not use any cryptography, we adopt the following simple signature:

$$\Sigma_{pair} = \{\mathsf{ok}, \mathsf{fail}, \mathsf{fst}, \mathsf{snd}, \mathsf{pair}\}$$

where ok and fail are constants ; fst and snd are unary functions and pair is a binary function. The term $\mathsf{pair}(m_1, m_2)$ represents the concatenation of two messages m_1 and m_2, while fst and snd represent the projections on the first and second component respectively. The set of terms $T(\mathcal{X}, \mathcal{N})$ is formally defined by the following grammar:

$$t, t_1, t_2, \cdots ::= x \mid n \mid \mathsf{pair}(t_1, t_2) \mid \mathsf{fst}(t) \mid \mathsf{snd}(t) \qquad x \in \mathcal{X}, n \in \mathcal{N}.$$

We write $\{^{M_1}/_{x_1}, \ldots, ^{M_n}/_{x_n}\}$ for the *substitution* that replaces the variables x_i by the terms M_i. The application of a substitution σ to a term N is denoted $N\sigma$. A term is *ground* if it does not contain variables. We also use the following notations: $\langle u_1, \ldots, u_n \rangle$ for $\mathsf{pair}(u_1, \mathsf{pair}(\ldots, \mathsf{pair}(u_{n-1}, u_n)))$ and $\Pi_i^n(u)$ for retrieving the i^{th} element of a sequence of n elements: $\Pi_i^n(u) = \mathsf{fst}(\mathsf{snd}^{i-1}(u))$ for $i < n$ and $\Pi_n^n(u) = \mathsf{snd}^{n-1}(u)$. In particular, $\Pi_i^n(\langle u_1, \ldots, u_n \rangle) = u_i$. We also write $x \in_n y$ for $[x = \Pi_1^n(y)] \vee \cdots \vee [x = \Pi_n^n(y)]$, that is, if x is one of the elements of the sequence y.

The properties of the pair are modeled by an equational theory E_{pair} that states that it is possible to retrieve the two elements of a pair:

$$\mathsf{fst}(\mathsf{pair}(x, y)) = x \qquad \mathsf{snd}(\mathsf{pair}(x, y)) = y.$$

We consider equality modulo this equational theory, that is, equality of terms is the smallest equivalence relation induced by E_{pair}, closed under application of function symbols, substitution of terms for variables and bijective renaming of names. We write $M == N$ for the syntactic equality.

Protocols themselves are modeled by *processes* and *extended processes*, as defined in Figure 4. Processes contain the basic operators to model a small programming language: 0 represents a process which does nothing, the parallel composition of the two processes

$\phi, \psi ::=$	formulae
$M = N \mid M \neq N \mid \phi \wedge \psi \mid \phi \vee \psi$	
$P, Q, R ::=$	(plain) processes
0	null process
$P \mid Q$	parallel composition
$!P$	replication
$\nu n.P$	name restriction
if ϕ then P else Q	conditional
$u(x).P$	message input
$\overline{u}\langle M\rangle.P$	message output
$\mathsf{event}(M).P$	event
$A, B, C ::=$	extended processes
P	plain process
$A \mid B$	parallel composition
$\nu n.A$	name restriction
$\nu x.A$	variable restriction
$\{^{M}/_{x}\}$	active substitution

Fig. 4. Syntax for processes

P and Q is denoted by $P \mid Q$, while $!P$ denotes the unbounded replication of P (that is, the unbounded parallel composition of P with itself). The process $\nu n.P$ creates a fresh name n and behaves like P. Tests are modeled by the process if ϕ then P else Q, which behaves like P if ϕ holds and like Q otherwise. Note that like in [6], we extend the applied pi-calculus by letting conditional branches now depend on formulae instead of just equality of terms. Process $u(x).P$ inputs some message (stored in the variable x) on channel u and then behaves like P while $\overline{u}\langle M\rangle.P$ outputs M on channel u and then behaves like P. $\mathsf{event}(M).P$ behaves like P, the event is there to record what happens during the execution of the protocol and is typically used to express properties. We write $\nu\tilde{u}$ for the (possibly empty) series of pairwise-distinct binders $\nu u_1. \ldots .\nu u_n$. The active substitution $\{^{M}/_{x}\}$ can replace the variable x by the term M in every process it comes into contact with and this behavior can be controlled by restriction, in particular, the process $\nu x\left(\{^{M}/_{x}\} \mid P\right)$ corresponds exactly to let $x = M$ in P.

Example 1. Let $P(a, b) = c(x).c(y).(\overline{c}\langle\langle x, a\rangle\rangle \mid \overline{c}\langle\langle y, b\rangle\rangle)$. This process waits for two inputs x and y on channel c then performs two outputs, $\langle x, a\rangle$, $\langle y, b\rangle$, in a non-deterministic order, on the same channel.

The *scope* of names and variables are delimited by binders $u(x)$ and νu. The different sets of bound names, bound variables, free names and free variables are respectively written $\mathsf{bn}(A)$, $\mathsf{bv}(A)$, $\mathsf{fn}(A)$ and $\mathsf{fv}(A)$. Occasionally, we write $\mathsf{fn}(M)$ (respectively $\mathsf{fv}(M)$) for the set of names (respectively variables) which appear in term M. An extended process is *closed* if all its variables are either bound or defined by an active substitution. An *context* $C\,[\_]$ is an extended process with a hole.

A *frame* is an extended process built up from the null process 0 and active substitutions composed by parallel composition and restriction. The *domain* of a frame φ, denoted $\mathsf{dom}(\varphi)$, is the set of variables for which φ contains an active substitution $\{^{M}/_{x}\}$ such that x is not under restriction. Every extended process A can be mapped to a frame $\varphi(A)$ by replacing every plain process in A with 0.

4.2 Semantics

The operational semantics of processes in the applied pi-calculus is defined by three relations: *structural equivalence* ($\equiv$), *internal reduction* ($\rightarrow$) and *labelled reduction* ($\xrightarrow{\alpha}$), formally defined in [1]. Structural equivalence is the smallest equivalence relation on extended processes that is closed under application of evaluation contexts, by α-conversion of bounded names and bounded variables. Internal reductions represent evaluation of condition and internal communication between processes while labelled reductions represent communication with the environment. For example, the input and output rules are represented by the following two rules:

$$\begin{array}{ll} (\text{In}) & c(x).P \xrightarrow{c(M)} P\{^{M}/_{x}\} \\[1ex] (\text{Out-Atom}) & \overline{c}\langle u\rangle.P \xrightarrow{\overline{c}\langle u\rangle} P \end{array}$$

Example 2. Let us consider the process $P(a, b)$ defined in Example 1 and the process $Q = \nu r.\overline{c}\langle r\rangle.\overline{c}\langle r\rangle$ that generates a random r and send it twice. A possible sequence of transitions for the process $P(a, b) \mid Q$ is:

$$\begin{aligned} &P(a,b) \mid Q \xrightarrow{\nu r_1.\overline{c}\langle r_1\rangle} P(a,v) \mid \nu r.\overline{c}\langle r\rangle \mid \{^{r}/_{r_1}\} \xrightarrow{\nu r_2.\overline{c}\langle r_2\rangle} P(a,b) \mid \{^{r}/_{r_1},{}^{r}/_{r_2}\} \\ &\xrightarrow{c(r_1)} c(y).(\overline{c}\langle\langle r,a\rangle\rangle \mid \overline{c}\langle\langle y,b\rangle\rangle) \mid \{^{r}/_{r_1},{}^{r}/_{r_2}\} \xrightarrow{c(r_2)} \overline{c}\langle\langle r,a\rangle\rangle \mid \overline{c}\langle\langle r,b\rangle\rangle \mid \{^{r}/_{r_1},{}^{r}/_{r_2}\} \\ &\xrightarrow{\nu y_1.\overline{c}\langle y_1\rangle} \overline{c}\langle\langle y,b\rangle\rangle \mid \{^{r}/_{r_1},{}^{r}/_{r_2},{}^{\langle r,a\rangle}/_{y_1}\} \xrightarrow{\nu y_2.\overline{c}\langle y_2\rangle} \{^{r}/_{r_1},{}^{r}/_{r_2},{}^{\langle r,a\rangle}/_{y_1},{}^{\langle r,b\rangle}/_{y_2}\}. \end{aligned}$$

At the end of the execution, the process is reduced to a frame that contains the terms emitted by the initial process.

Privacy properties are often stated as equivalence relations [8]. Intuitively, if a protocol preserves ballot secrecy, an attacker should not make a distinction between a scenario where a voter votes 0 from a scenario where the voter votes 1. The applied pi-calculus comes with the notion of *observational equivalence*, which formally defines what it means for two processes to be indistinguishable for any attacker. Since observational equivalence has been shown to coincide [1,17] with labelled bisimilarity, which is easier to reason with, we adopt the latter in this paper. Labelled bisimilarity intuitively states that processes should be bisimilar and send indistinguishable messages. In our context, given that the only primitive we consider is pairing, two sequences of messages are indistinguishable to an attacker (formally defined as static equivalence [1]) if and only if they are equal. We therefore present here a simplified version of labelled bisimilarity, which is labelled bisimilarity for the special case of pairing.

Definition 1 (Labelled bisimilarity). *Labelled bisimilarity ($\approx_l$) is the largest symmetric relation $\mathcal{R}$ on closed extended processes such that $A\mathcal{R}B$ implies:*

1. *$\varphi(A) = \varphi(B)$;*
2. *if $A \rightarrow A'$, then $B \rightarrow^* B'$ and $A'\mathcal{R}B'$ for some B';*
3. *if $A \xrightarrow{\alpha} A'$ such that $\mathsf{fv}(\alpha) \subseteq \mathsf{dom}(A)$ and $\mathsf{bn}(\alpha) \cap \mathsf{fn}(B) = \emptyset$, then $B \rightarrow^* \xrightarrow{\alpha} \rightarrow^* B'$ and $A'\mathcal{R}B'$ for some B'.*

Example 3. Let us consider $A = P(a, b) \mid Q$ and $B = P(b, a) \mid Q$. Is $A \approx_l B$? Let us consider the same evolution as in Example 2 except that $c(r_1)$ and $c(r_2)$ are replaced by $c(M)$ and $c(N)$ which represents an action of the intruder, replacing what is sent by Q by something of her choice. In that case, we will have :

$$\varphi(A) = \{^r/_{r_1}, ^r/_{r_2}, ^{\langle M,a\rangle}/_{y_1}, ^{\langle N,b\rangle}/_{y_2}\} \text{ and } \varphi(B) = \{^r/_{r_1}, ^r/_{r_2}, ^{\langle M,b\rangle}/_{y_1}, ^{\langle N,a\rangle}/_{y_2}\}.$$

Since $\varphi(A) \neq \varphi(B)$ we have that $A \not\approx_l B$.

4.3 Modeling Protocols in Applied pi-Calculus

We provide a formal specification of the two last variants of the CNRS voting system, in the applied pi-calculus. We do not describe the formal model of the initial voting system since it does not ensure ballot secrecy nor vote correctness.

We model the communications of the ballot box with the voters and the screen by secure channels (resp. $\mathsf{c_i}$ and $\mathsf{c_B}$). These channels may be controlled by the adversary when the ballot box is corrupted. The voters and the assessor look at the screen. This communication cannot be altered and is modeled by an authenticated channel $\mathsf{c_{eyes}}$. The assessor also communicates with each voter to check that the voter found his/her ballot on the screen. This is again modeled by an authenticated channel $\mathsf{c_{A_i}}$ since we assume that voters cannot be physically impersonated. The channel connections are summarized in Figure 5.

Remark 1. The applied-pi calculus provides an easy way to model both public and secure channel. Public channels are simply modeled by unrestricted names: the attacker can both read and send messages. Secure channels are modeled by restricted names: the attacker cannot read nor send any message on these channels. In contrast, an attacker may read authenticated channels but only authorized users may send messages on them. Since the applied pi-calculus does not provide us with a primitive for authenticated channels, we model authenticated channel by a secure channel, except that a copy of each emission is sent first on a public channel. In particular, we use the notation $\overline{\overline{c}}\langle M\rangle$ for $\overline{c_p}\langle M\rangle.\overline{c}\langle M\rangle$ with c_p a public channel.

Remark 2. The role of the individual voting device is limited: it simply receives the vote from the voter and transmit it to the Ballot Box. W.l.o.g and for simplicity, we identify the voter and her individual device in the model of the voting systems.

Model of $\mathsf{F2FV^2}$. The process for the voter is parametrized by the number n of voters, its secure channel with the ballot box c, its authenticated channel with the screen (c_e) and the auditor (c_a), the public channel c_p and its vote v.

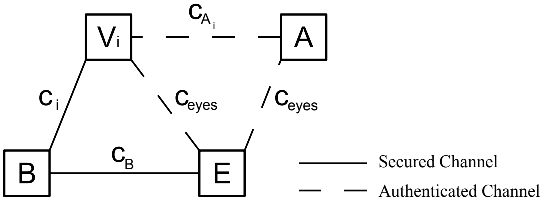


Fig. 5. Players of the Protocol

```
$V_n(c, c_e, c_a, c_p, v) =$
        $\nu k \,.\, c(x) \,.$                      % Creates fresh nonce and waits for input on $c$.
        $\overline{c}\langle\langle x, k, v\rangle\rangle \,.$              % Sends ballot on $c$ to the ballot box.
        $c_e(y) \,.$                          % Waits for input on $c_e$ (results on the screen).
        if $\langle x, k, v\rangle \in_n y$                   % Checks his vote.
        then $\overline{\overline{c_a}}\langle \mathsf{ok}\rangle$ else $\overline{\overline{c_a}}\langle \mathsf{fail}\rangle$   % Sends result on $c_a$ to the assessor.
```

The process for the ballot box is parametrized by the number n of voters, the secure channels $c_v^1, \dots, c_v^n$ with each voter and its secure channel with the screen c_{be}.

```
$B_n(c_v^1, \dots, c_v^n, c_b) =$
        $\nu r_1, \dots, r_n \,.$                                  % Creates fresh randomness.
        $\overline{c_v^1}\langle r_1\rangle \,.\, \dots \,.\, \overline{c_v^n}\langle r_n\rangle \,.$              % Sends randomness to voters.
        $c_v^1(y_1) \,.\, \dots \,.\, c_v^n(y_n) \,.$               % Waits for inputs of ballots.
        $(\overline{c_b}\langle y_1\rangle \mid \cdots \mid \overline{c_b}\langle y_n\rangle)$          % Sends ballots in random order to $E$.
```

The screen is modeled by a process E_n that simply broadcasts the result given by B_n. It is parametrized by the number n of voters, the authenticated channels c_e with each voter, the secure channel with the bulletin box c_b, and the public channel c_p.

```
$E_n(c_b, c_e, c_p) =$
        $c_b(t_1) \,.\, \dots \,.\, c_b(t_n) \,.$                 % Waits for votes from ballot box.
        let $r = \langle t_1, \dots, t_n\rangle$ in
        $\overline{c_p}\langle r\rangle \,.\, (!\, \overline{c_e}\langle r\rangle)$                   % Displays info for all the boardroom.
```

The last role is the role of the assessor. It is modeled by a process A_n that waits for the result displayed by the screen and the confirmation of the voters. Then it verifies the outcome and validates the election if everything is correct. The process A_n is parametrized by the number n of voters, the authenticated channels $c_a^1, \dots, c_a^n$ with each voter, the secure channel with the screen c_e, and the public channel c_p.

```
$A_n(c_e, c_a^1, \dots, c_a^n, c_p) =$
        $c_e(z') \,.$                                % Waits to see result on the screen.
        $c_a^1(z_1) \,.\, \dots \,.\, c_a^n(z_n) \,.$               % Waits for decision of voters.
        if $\Psi_n(z', z_1, \dots, z_n)$               % Checks if everything is fine.
        then $\overline{c_p}\langle \mathsf{ok}\rangle$ else $\overline{c_p}\langle \mathsf{fail}\rangle$     % Sends confirmation or rejection.
```

where $\Psi_n(p', p_1, \ldots, p_n) = (\bigwedge_{i=1}^{n} p_i = \mathsf{ok}) \wedge (p' = \langle \Pi_1^n(p'), \Pi_2^n(p'), \ldots, \Pi_n^n(p')\rangle)$. The test Ψ_n ensures that each voter approved the vote ($p_i = \mathsf{ok}$) and that the result contains as many ballots than the number of voters.

Finally the system F2FV^2 is represented by the voter's role V_n and the voting context:

$$P_n^2\,[\,\_\,] = \nu\,\widetilde{\omega}.\,[\,\_\,|B_n(\mathsf{c_1}, \ldots, \mathsf{c_n}, \mathsf{c_B})|E_n(\mathsf{c_B}, \mathsf{c_{eyes}}, \mathsf{c_{out}})|A_n(\mathsf{c_{eyes}}, \mathsf{c_{A_1}}, \ldots, \mathsf{c_{A_n}}, \mathsf{c_{out}})]$$

where $\widetilde{\omega} = (\mathsf{c_1}, \ldots, \mathsf{c_n}, \mathsf{c_{A_1}}, \ldots, \mathsf{c_{A_n}}, \mathsf{c_B}, \mathsf{c_{eyes}})$ are restricted channels ($\mathsf{c_{out}}$ is public).

Model of the Protocol F2FV^3. The third protocol only differs from the second one by the fact that the ballot box does not generate any randomness. Therefore, the models of the screen and of the assessor are unchanged. The voter and ballot box models are modified as follows.

$$\begin{array}{l} V_n'(c, c_e, c_a, c_p, v) = \\ \qquad \nu k\;.\;\overline{c}\langle\langle k, v\rangle\rangle\;.\;c_e(x)\;. \\ \qquad \text{if } \langle k, v\rangle \in_n x \text{ then } \overline{\overline{c_a}}\langle \mathsf{ok}\rangle \text{ else } \overline{\overline{c_a}}\langle \mathsf{fail}\rangle \end{array}$$

$$\begin{array}{l} B_n'(c_v^1, \ldots, c_v^n, c_b) = \\ \qquad c_v^1(y_1)\;.\;\ldots\;.\;c_v^n(y_n)\;. \\ \qquad (\overline{c_b}\langle y_1\rangle \mid \cdots \mid \overline{c_b}\langle y_n\rangle) \end{array}$$

The system F2FV^3 without the voters is represented by the voter's role V_n' and the voting context:

$$P_n^3\,[\,\_\,] = \nu\,\widetilde{\omega}.\,[\,\_\,|B_n'(\mathsf{c_1}, \ldots, \mathsf{c_n}, \mathsf{c_B})|E_n(\mathsf{c_B}, \mathsf{c_{eyes}}, \mathsf{c_{out}})|A_n(\mathsf{c_{eyes}}, \mathsf{c_{A_1}}, \ldots, \mathsf{c_{A_n}}, \mathsf{c_{out}})]$$

where $\widetilde{\omega} = (\mathsf{c_1}, \ldots, \mathsf{c_n}, \mathsf{c_{A_1}}, \ldots, \mathsf{c_{A_n}}, \mathsf{c_B}, \mathsf{c_{eyes}})$ are restricted channels.

5 Security Properties

We study two crucial properties for voting systems: ballot secrecy and vote correctness. We consider two cases depending on whether the ballot box is corrupted or not. We always assume the screen to be honest. This is however not a limitation. Indeed, requiring the screen to be honest reflects the fact that everyone sees the same screen, which is always the case for people in the same room.

5.1 Ballot Secrecy

Formalizing ballot secrecy may be tricky. For example, even a good voting system reveals how anyone voted in case of unanimity. Early definitions of privacy appear for example in [3]. In what follows, we use a well established definition of ballot secrecy that has been formalized in terms of equivalence by Delaune, Kremer and Ryan in [8]. Several other definitions of privacy have been proposed (see e.g. [15,4]), which measure the fact that the attacker may learn some information, even if he does not know how a certain voter voted.

A protocol with voting process $V(v, id)$ and authority process A preserves *ballot secrecy* if an attacker cannot distinguish when votes are swapped, i.e. it cannot distinguish when a voter a_1 votes v_1 and a_2 votes v_2 from the case where a_1 votes v_2 and a_2 votes v_1. This is formally specified by :

$$\nu\tilde{n}.\,(A \mid V\{^{v_2}/_x, ^{a_1}/_y\} \mid V\{^{v_1}/_x, ^{a_2}/_y\}) \approx_l \nu\tilde{n}.\,(A \mid V\{^{v_1}/_x, ^{a_1}/_y\} \mid V\{^{v_2}/_x, ^{a_2}/_y\})$$

where $\tilde{n}$ represents the data (keys, nonces, channels, ...) initially shared between the authority and the voters.

Ballot Secrecy for Voting Protocol $\mathsf{F2FV}^2$. The voting protocol $\mathsf{F2FV}^2$ preserves ballot secrecy, even when all but two voters are dishonest, provided that the ballot box, the screen and the assessor are honest. For the sake of clarity, we use the following notation for the i^{th} voter: $V^i(v) = V_n(\mathsf{c_i}, \mathsf{c_{eyes}}, \mathsf{c_{A_i}}, \mathsf{c_{out}}, v)$.

Theorem 1. *Let $n \in \mathbb{N}$, let (P_n^2, V_n) be the process specification for n voters of the voting protocol* $\mathsf{F2FV}^2$ *as defined in Section 3.2, and let a, b be two names. Then*

$$P_n^2\left[V^1(a) \mid V^2(b)\right] \approx_l P_n^2\left[V^1(b) \mid V^2(a)\right]$$

Proof sketch: The proof of Theorem 1 consists in two main steps. First we build a relation $\mathcal{R}$ such that

$$P_n^2\left[V^1(a) \mid V^2(b)\right] \ \mathcal{R}\ P_n^2\left[V^1(b) \mid V^2(a)\right]$$

and such that for any two processes $P\ \mathcal{R}\ Q$, any move of P can be matched by a move of Q such that the resulting processes remain in relation. This amounts to characterizing all possible successors of $P_n^2\left[V^1(a) \mid V^2(b)\right]$ and $P_n^2\left[V^1(b) \mid V^2(a)\right]$. The second step of the proof consists in showing that the sequences of messages observed by the attacker are equal (due to the shuffle performed by the ballot box).

Ballot Secrecy for Voting Protocol $\mathsf{F2FV}^3$. Similarly, the voting protocol $\mathsf{F2FV}^3$ preserves ballot secrecy, even when all but two voters are dishonest, provided that the ballot box, the screen and the assessor are honest.

Theorem 2. *Let $n \in \mathbb{N}$, let (P_n^3, V_n') be the process specification for n voters of the voting protocol* $\mathsf{F2FV}^3$ *as defined in Section 3.3, and let a, b be two names. Then*

$$P_n^3\left[{V'}^1(a) \mid {V'}^2(b)\right] \approx_l P_n^3\left[{V'}^1(b) \mid {V'}^2(a)\right]$$

The proof of Theorem 2 is adapted from the proof of Theorem 1.

5.2 Vote Correctness

We define vote correctness as the fact that the election result should contain the votes of the honest voters. Formally, we assume that the voting protocol records the published outcome of the election t in an event $\mathsf{event}(t)$.

Definition 2 (Correctness property). *Let n be the number of registered voters, and m be the number of honest voters. Let $v_1, \ldots, v_m \in \mathcal{N}$ be the votes of the honest voters. Let $V^1, \ldots, V^m$ be the processes representing the honest voters. Each V^i is parametrized by its vote v_i. Let P_n be a context representing the voting system, besides the honest voters. We say that a voting specification $(P_n, \tilde{V})$ satisfies* vote correctness *if for every $v_1, \ldots, v_m$, for every execution of the protocol leading to the validation of a result t_r, i.e. of the form*

$$P_n[V^1(v_1)|\ldots|V^m(v_m)] \to^* \nu\tilde{n}\cdot(\mathsf{event}(t_r)\cdot Q \mid Q')$$

for some names $\tilde{n}$ and processes Q, Q', then there exist votes $v_{m+1}, \ldots, v_n$ and a permutation τ of $[\![1, n]\!]$ such that $t_r = \langle v_{\tau(1)}, \ldots, v_{\tau(n)}\rangle$, that is, the outcome of the election contains all the honest votes plus some dishonest ones.

To express vote correctness in the context of the CNRS Section 07 voting system, we simply add an event that records the tally, at the end of the process specification of the assessor (see Appendix for the corresponding modified process A'_n). We show vote correctness for a strong corruption scenario, where even the ballot box is corrupted. Formally, we consider the following context that represents the three voting systems, the only difference between the systems now lying in the definition of voters.

$$P_n{}'\,[\,\_\,] = \nu\,\widetilde{\omega}.\,[\,\_ \mid E_n(\mathsf{c_B}, \mathsf{c_{eyes}}, \mathsf{c_{out}}) \mid A'_n(\mathsf{c_{eyes}}, \mathsf{c_{A_1}}, \ldots, \mathsf{c_{A_n}}, \mathsf{c_{out}})]$$

where $\widetilde{\omega} = (\mathsf{c_{A_1}}, \ldots, \mathsf{c_{A_n}}, \mathsf{c_{eyes}})$, which means that the intruder has access in this scenario to channels $\mathsf{c_1}, \ldots, \mathsf{c_n}$ and $\mathsf{c_B}$ in addition to $\mathsf{c_{out}}$.

To illustrate the correctness property, let first show that $\mathsf{F2FV}^1$ does not satisfy vote correctness when the ballot box is corrupted. First, we introduce $\hat{V}$ the process of an honest voter in $\mathsf{F2FV}^1$:

$$\hat{V}(c, c_e, c_a, c_p, v) = c(x)\ .\ \overline{c}\langle\langle x, v\rangle\rangle\ .\ c_e(y)\ .\ \text{if } \langle x, v\rangle \in_n y \text{ then } \overline{\overline{c_a}}\langle\mathsf{ok}\rangle \text{ else } \overline{\overline{c_a}}\langle\mathsf{fail}\rangle$$

Let $\hat{V}^i = \hat{V}\{^{\mathsf{c_i}}/_{c}, ^{\mathsf{c_{eyes}}}/_{c_e}, ^{\mathsf{c_{A_i}}}/_{c_a}, ^{\mathsf{c_{out}}}/_{c_p}\}$. It represents the i-th honest voter. Suppose now, that the first m honest voters cast the some vote: $\forall i \in [\![1, m]\!]$, $v_i = v$. We show how the attack described in Section 3.1 is reflected. Each honest voter receives the same random number r:

$$P'_n[\hat{V}^1(v_1) \mid \cdots \mid \hat{V}^m(v_m)] \xrightarrow{\forall i\in[\![1,m]\!],\ \overline{c_i}\langle r\rangle} P'_n[\hat{V}^1_r(v_1) \mid \cdots \mid \hat{V}^m_r(v_m)]$$

where $\hat{V}^i_r(v_i) = \overline{\mathsf{c_i}}\langle\langle r, v_i\rangle\rangle\ .\ \mathsf{c_{eyes}}(y)\ .\ \text{if } \langle r, v_i\rangle \in_n y_i \text{ then } \overline{\overline{\mathsf{c_{A_i}}}}\langle\mathsf{ok}\rangle \text{ else } \overline{\overline{\mathsf{c_{A_i}}}}\langle\mathsf{fail}\rangle$. Then, the honest voters output their vote on channels $\mathsf{c_1}, \ldots, \mathsf{c_m}$ which will always be $\langle r, v\rangle$.

$$P'_n[\hat{V}^1_r(v_1) \mid \cdots \mid \hat{V}^m_r(v_m)] \xrightarrow{\forall i\in[\![1,m]\!],\ \overline{c_i}\langle\langle r, v_i\rangle\rangle} P'_n[\hat{V}^1_e(v_1) \mid \cdots \mid \hat{V}^m_e(v_m)]$$

where $\hat{V}^i_e(v_i) = \mathsf{c_{eyes}}(y)\ .\ \text{if } \langle r, v_i\rangle \in_n y_i \text{ then } \overline{\overline{\mathsf{c_{A_i}}}}\langle\mathsf{ok}\rangle \text{ else } \overline{\overline{\mathsf{c_{A_i}}}}\langle\mathsf{fail}\rangle$. Corrupted voters also submit their votes (which is transparent in transitions) and we move to the next phase: the corrupted ballot box just has to output one of the honest votes to the screen and $n-1$ other votes. Thus, the final tally t_r showed by the screen will contain only one $\langle r, v\rangle$ but each honest voters will send ok to the assessor since their test will succeed anyway. In that case, we would have $P'_n[\hat{V}^1(v_1)|\ldots|\hat{V}^m(v_m)] \to^* \nu\tilde{n}\cdot\mathsf{event}(t_r)$ for some $\tilde{n}$, but, clearly, t_r is not satisfying the property of the Definition 2 since it only contains one vote v instead of m votes v.

In contrast, the two voting systems $\mathsf{F2FV}^2$ and $\mathsf{F2FV}^3$ satisfy vote correctness, even when the ballot box is corrupted, assuming that the voters check that their ballots appear on the screen.

Table 1. Results for the F2FV^1,F2FV^2, and F2FV^3 protocols. A ✓ indicates provable security while × indicates an attack. We assume an arbitrary number of dishonest voters.

RESULTS	Privacy			Correctness		
Corr. Players System	None	Ballot Box	Assessor	None	Ballot Box	Assessor
F2FV^1	✓	×	✓	✓	×	×
F2FV^2	✓	×	✓	✓	✓	×
F2FV^3	✓	×	✓	✓	✓	×

Theorem 3. *The voting specifications* (P'_n, V) *and* (P'_n, V') *satisfy vote correctness.*

Proof sketch. The assessor records the result of the election in an event only if $\Psi_n(p', p_1, \ldots, p_n)$ holds. This formula intuitively represents the fact that every voter has told to the assessor that his ballot was included in the tally, and that the number of ballots in the tally matches the number of voters, i.e. n. Using this information and the fact that each honest voter has generated a random nonce uniquely identifying his ballot, we can show that the voting specifications satisfy vote correctness.

Correctness requires that at least one person in the room checks that no one has complained and that the number of displayed ballots correspond to the number of voters. If no one performs these checks then there is no honest assessor and correctness is no longer guaranteed.

A summary of our findings is displayed on Table 1. The proofs of correctness of F2FV^2 and F2FV^3 in the honest case follow from the proofs in the dishonest case. Privacy is not affected by a corrupted assessor as it actually only performs public verification. So its corruption does not provide any extra power to the attacker. Privacy and correctness for F2FV^1 (in the honest case) follow from the proofs for F2FV^2.

6 Discussion

We believe that the voting system proposed by the CNRS Section 07 committee for boardroom meetings is an interesting protocol that improves over existing electronic devices. We have analyzed the security of three possible versions, discovering some interesting flaws. We think that the two last versions are adequate since they both preserve ballot secrecy and vote correctness. The choice between the two versions depends on the desired compromise between ballot secrecy and vote correctness: the second version ensures better correctness but less privacy since the randomness generated by the ballot box may leak the identity of the voters. Conversely, the third system offers better privacy but slightly less assurance about vote correctness, in case the voters do not use proper random identifiers.

In both cases, vote correctness is guaranteed as soon as:

- Voters really use (unpredictable) random numbers. In practice, voters could print (privately and before the meeting) a list of random numbers that they would use at their will (erasing a number once used). This list of random numbers could typically be generated using a computer. Alternatively, voters may also bring dice to the meeting.
- Each voter casts a vote (possibly blank or null) and checks that his vote (and associated randomness) appears on the screen.

Correctness does not require any trust on the devices while privacy does. This is unavoidable unless the communication between the voters on the ballot box would be anonymized, which would require a much heavier infrastructure. Note that the system is not fair if the ballot box is compromised since dishonest voters may then wait for honest voters to cast their votes, before making their own decision.

In this paper, we have focused on ballot secrecy and vote correctness. As future work, we plan to study stronger notions of privacy. Clearly, the voting system is not coercion resistant. Indeed, an attacker may provide a voter with a list of random numbers, that he should use in a precise order, allowing the attacker to control the votes. However, we believe these systems ensure some form of receipt-freeness, assuming the attacker is given access to the screen only after the election is over but cannot interact with voters before nor during the election.

A weakness of the system relies in the fact that a voter may force to re-run an election by (wrongly) claiming that her vote does not appear on the screen. As already mentioned in Section 3.4, this is mitigated by the fact that the voter could then be blamed if this happens to often. This also means that an honest voter could be blamed if a dishonest Ballot Box intentionally removes her ballot at each turn. It would be interesting to devise a mechanism to mitigate this issue.

Acknowledgment. We would like to thank the anonymous reviewers for their numerous remarks and propositions that helped us to improve the paper.

References

1. Abadi, M., Fournet, C.: Mobile values, new names, and secure communication. In: 28th ACM Symp. on Principles of Programming Languages (POPL 2001), pp. 104–115 (2001)
2. Adida, B.: Helios: web-based open-audit voting. In: 17th Conference on Security Symposium, SS 2008, pp. 335–348. USENIX Association (2008)
3. Benaloh, J., Tuinstra, D.: Receipt-free secret-ballot elections. In: Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC 1994), pp. 544–553. ACM (1994)
4. Bernhard, D., Cortier, V., Pereira, O., Warinschi, B.: Measuring vote privacy, revisited. In: 19th ACM Conference on Computer and Communications Security (CCS 2012), Raleigh, USA. ACM (October 2012)
5. Clarkson, M.R., Chong, S., Myers, A.C.: Civitas: Toward a secure voting system. In: 2008 IEEE Symposium on Security and Privacy, pp. 354–368 (2008)
6. Cortier, V., Smyth, B.: Attacking and fixing Helios: An analysis of ballot secrecy. In: 24th IEEE Computer Security Foundations Symposium (CSF 2011), pp. 297–311 (2011)

7. Cortier, V., Wiedling, C.: A formal analysis of the norwegian E-voting protocol. In: Degano, P., Guttman, J.D. (eds.) POST 2012. LNCS, vol. 7215, pp. 109–128. Springer, Heidelberg (2012)
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. Feldman, A., Halderman, A., Felten, E.: Security Analysis of the Diebold AccuVote-TS Voting Machine. In: 2007 USENIX/ACCURATE Electronic Voting Technology Workshop, EVT 2007 (2007)
10. Fujioka, A., Okamoto, T., Ohta, K.: A practical secret voting scheme for large scale elections. In: Zheng, Y., Seberry, J. (eds.) AUSCRYPT 1992. LNCS, vol. 718, pp. 244–251. Springer, Heidelberg (1993)
11. Groth, J.: Efficient maximal privacy in boardroom voting and anonymous broadcast. In: Juels, A. (ed.) FC 2004. LNCS, vol. 3110, pp. 90–104. Springer, Heidelberg (2004)
12. Hao, F., Ryan, P.Y.A., Zielinski, P.: Anonymous voting by two-round public discussion. IET Information Security 4(2), 62–67 (2010)
13. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. 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. 37–63. Springer, Heidelberg (2010)
14. Kremer, S., Ryan, M.D., 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)
15. Küsters, R., Truderung, T., Vogt, A.: Verifiability, Privacy, and Coercion-Resistance: New Insights from a Case Study. In: IEEE Symposium on Security and Privacy (S&P 2011), pp. 538–553. IEEE Computer Society (2011)
16. Küsters, R., Truderung, T., Vogt, A.: Clash Attacks on the Verifiability of E-Voting Systems. In: IEEE Symposium on Security and Privacy (S&P 2012), pp. 395–409. IEEE Computer Society (2012)
17. Liu, J.: A proof of coincidence of labeled bisimilerity and observational equivalence in applied pi calculus. Technical report (2011)
18. Schneier, B.: Applied Cryptography, ch. 6. John Wiley & Sons (1996)
19. Wolchok, S., Wustrow, E., Halderman, J.A., Prasad, H.K., Kankipati, A., Sakhamuri, S.K., Yagati, V., Gonggrijp, R.: Security analysis of India's electronic voting machines. In: 17th ACM Conference on Computer and Communications Security, CCS 2010 (2010)