# A formal analysis of the Norwegian e-voting protocol[⋆]

Véronique Cortier and Cyrille Wiedling

LORIA - CNRS, Nancy, France

**Abstract.** Norway has used e-voting in its last political election in September 2011, with more than 25 000 voters using the e-voting option. The underlying protocol is a new protocol designed by the ERGO group, involving several actors (a bulletin box but also a receipt generator, a decryption service, and an auditor). Of course, trusting the correctness and security of e-voting protocols is crucial in that context. Formal definitions of properties such as privacy, coercion-resistance or verifiability have been recently proposed, based on equivalence properties.
In this paper, we propose a formal analysis of the protocol used in Norway, w.r.t. privacy, considering several corruption scenarios. Part of this study has conducted using the ProVerif tool, on a simplified model.

**Keywords:** e-voting, privacy, formal methods

## 1 Introduction

Electronic voting protocols promise a convenient, efficient and reliable way for collecting and tallying the votes, avoiding for example usual human errors when counting. It is used or have been used for political elections in several countries like e.g. USA, Estonia, Switzerland and recently Norway, at least in trials. However, the recent history has shown that these systems are highly vulnerable to attacks. For example, the Diebold machines as well as the electronic machines used in India have been attacked [13,24]. Consequently, the use of electronic voting raises many ethical and political issues. For example, the German Federal Constitutional Court decided on 3 March 2009 that electronic voting used for the last 10 years was unconstitutional [1].

There is therefore a pressing need for a rigorous analysis of the security of e-voting protocols. A first step towards the security analysis of e-voting protocols consists in precisely defining security w.r.t. e-voting. Formal definitions have been proposed for several key properties such as privacy, receipt-freeness, coercion resistance, or verifiability, most of them in terms of equivalence-based properties (see e.g. [12,17]). It is however difficult to formally analyse e-voting protocols for two main reasons. First there are very few tools that can check

[⋆] 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.

equivalence properties: ProVerif [5,6] is probably the only one but it does not really work in the context of e-voting (because it tries to show a stronger notion of equivalence, which is not fulfilled when checking for ballot secrecy). Some other very recent (yet preliminary) tools have been proposed such as Datep [8] or AKiSs [7]. However, the cryptographic primitives used in e-voting are rather complex and non standard and are typically not supported by existing tools.

In this paper, we study the protocol used in last September for political elections in Norway [2]. E-voting was proposed as trials in several municipalities and more than 25 000 voters did use e-voting to actually cast their vote. The protocol is publicly available [15] that has four main components: a Bulletin Box, a Decryption Service, and a Receipt Generator and an Auditor which aim at watching the Bulletin Box recording the votes. The resulting protocol is therefore complex, e.g. using El Gamal encryption in a non standard way. In [15], Gjøsteen describes the protocol and discusses its security. To our knowledge, there does not exist any security proof, even for the crucial property of vote privacy.

*Our contribution.* We conduct a formal analysis of the Norwegian protocol w.r.t. privacy. Our first contribution is the proposition of a formal model of the protocol in applied-pi calculus [3]. One particularity of the protocol is to distribute public keys $\mathsf{pk}(a_1)$, $\mathsf{pk}(a_2)$, and $\mathsf{pk}(a_3)$ for the three authorities, such that the corresponding private keys $a_1, a_2$, and $a_3$ verify the relation $a_1 + a_2 = a_3$, allowing one component (here the Bulletin Box) to re-encrypt messages. The protocol also makes use of signature, of zero-knowledge proofs, of blinding functions and coding functions. We have therefore proposed a new equational theory reflecting the unusual behavior of the primitives.

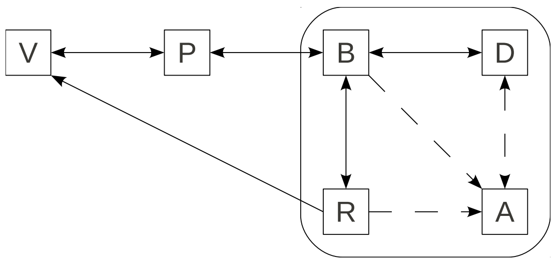
Our second contribution is a formal security proof of privacy, in the presence of arbitrarily many dishonest voters. Given the complexity of the equational theory (with e.g. four associative and commutative symbols), the resulting processes can clearly not be analyzed with existing tools, even ProVerif. We therefore proved privacy (expressed as an equivalence property) by hand. The proof happens to be quite technical. Its first step is rather standard and consists in guessing a relation such that the two initial processes and all their possible evolutions are in relation. The second step is more involved: it requires to prove equivalent an infinite number of frames, the frames representing all possible attacker knowledge. Indeed, unlike most previously analyzed protocols, the Norwegian protocol emits receipts for the voters, potentially providing extra information to the attacker. Proving static equivalence is also made difficult due to our equational theory (e.g. four associative and commutative symbols).

Our third contribution is an analysis of the protocol for further corruption scenarios, using the ProVerif tool in a simplified model (therefore possibly losing attacks). In conclusion, we did not find any attack, except when the bulletin box and the receipt generator or the decryption service alone (if no shuffling is made) are corrupted. These attacks are probably not surprising but we found interesting to make them explicit.

*Related Work.* [15] provides a discussion on the security of the Norwegian protocol but no security proof. We do not know any other study related to this protocol. Several other e-voting protocols have been studying using formal methods. The FOO [14], Okamoto [23] and Lee *et al.* [21] voting protocols have been analysed in [12]. Similarly, Helios has been recently proved secure both in a formal [10] and a computational [4] model. However, all these protocols were significantly simpler to analyse. The more complex Civitas protocol was analyzed in [18]. In contrast, the Norwegian protocol is both complex and fully deployed. There are also been studies of hybrid protocols (not fully electronic), such as Scantegrity [20] or ThreeBallot [19].

We informally describe the protocol in Section 2. The applied-pi calculus is briefly defined in Section 3. We then provide a formal modeling of the protocol in Section 4 and formally state and prove the privacy properties satisfied by the protocol in Section 5. The results obtained with ProVerif are described in Section 6. Concluding remarks can be found in Section 7. All the proofs are provided in a research report [11].

## 2 Norwegian E-Voting Protocol



Norwegian protocol features several players including four players representing the electronic poll's infrastructure : a ballot box (B), a receipt generator (R), a decryption service (D) and an auditor (A). Each voter (V) can log in using a computer (P) in order to submit his vote. Channels between computers (voters) and the ballot box are considered as authenticated channel, channels between infrastructure's player are untappable channels and channel between voters and receipt generator is a unidirectional out-of-band channel. (Example of SMS is given in [15].) The protocol can be divided in three phases : the setting phase, the submission phase, where voters submit their votes, and the counting phase, where ballots are counted and auditor verifies the correctness of the election.

### 2.1 Setting phase

Before the election, private keys $a_1$, $a_2$, and $a_3$ (such that $a_1 + a_2 = a_3$) are distributed over, respectively D, B, and R, while the corresponding public keys are made publicly available. The receipt generator R is assumed to have a signing key $\mathsf{id}_R$ which corresponding verification key is distributed to P. The voters are also assume to each have a signing key $\mathsf{id}_V$ with the corresponding verification key distributed to B. The bulletin board B is provided with a table $V \mapsto s_V$

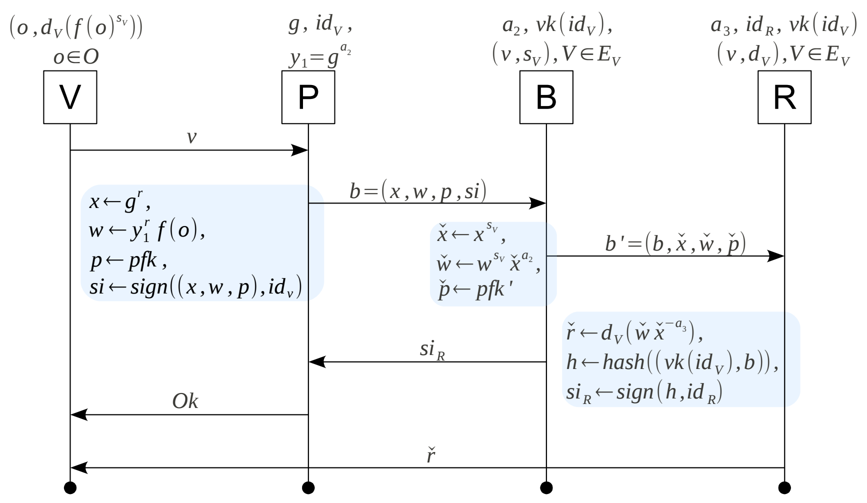

$(o, d_V(f(o)^{s_V}))$
$o \in O$
$g$, $id_V$,
$y_1 = g^{a_2}$
$a_2$, $vk(id_V)$,
$(v, s_V), V \in E_V$
$a_3$, $id_R$, $vk(id_V)$
$(v, d_V), V \in E_V$
V
P
B
R
$v$
$b = (x, w, p, si)$
$x \leftarrow g^r$,
$w \leftarrow y_1^r f(o)$,
$p \leftarrow pfk$,
$si \leftarrow sign((x, w, p), id_v)$
$\check{x} \leftarrow x^{s_V}$,
$\check{w} \leftarrow w^{s_V} \check{x}^{a_2}$,
$\check{p} \leftarrow pfk'$
$b' = (b, \check{x}, \check{w}, \check{p})$
$\check{r} \leftarrow d_V(\check{w}\check{x}^{-a_3})$,
$h \leftarrow hash((vk(id_V), b))$,
$si_R \leftarrow sign(h, id_R)$
$si_R$
$Ok$
$\check{r}$

**Fig. 1.** Submission of one vote.

with a blinding factor $s_V$ for each voter $V$. The receipt generator R is given a table $V \mapsto d_V$ with a permutation function $d_V$ for each voter $V$. Finally, each voter V is assumed to received by post a table where, for each voting option $o$ corresponds a precomputed receipt code $d_V(f(o)^{s_V})$ where $f$ is some encoding function for voting options.

## 2.2 Submission phase

The submission phase is depicted in Figure 1. We detail below the expected behavior of each participant.

*Voter (V).* Each voter tells his computer what voting option $o$ to submit and allows it to sign the corresponding ballot on his behalf. Then, he has to wait for an acceptance message coming from the computer and a receipt $\check{r}$ sent by the receipt generator through the out-of-band channel. Using the receipt, he verifies that the correct vote was submitted, that is, he checks that $\check{r} = d_V(f(o)^{s_V})$ by verifying that the receipt code $\check{r}$ indeed appears in the line associated to $o$.

*Computer (P).* Voter's computer encrypts voter's ballot with the public key $y_1$ using standard El Gamal encryption. The resulting ballot is $(g^r, y^r f(o))$. P also proves that the resulting ciphertext corresponds to the correct vote, by computing a standard signature proof of knowledge *pfk*. How *pfk* is computed exactly can be found in [15]. P also signs the ballot with $\mathsf{id}_V$ and sends it to the ballot box. It then waits for a confirmation $\mathsf{si}_R$ coming from the latter, which is a hash of the initial encrypted ballot, signed by the receipt generator. After checking this signature, the computer notifies the voter that his vote has been taken into account.

*Bulletin Box (B).* Receiving an encrypted and signed ballot $b$ from a computer, the ballot box checks first the correctness of signatures and proofs before re-encrypting with $a_2$ and blinding with $s_V$ the original encrypted ballot, also generating a proof $pfk'$, showing that its computation is correct. B then sends the new modified ballot $b'$ to the receipt generator. Once the ballot box receives a message $\mathsf{si}_R$ from the receipt generator, it simply checks that the receipt generator's signature is valid, and sends it to the computer.

*Receipt generator (R).* When receiving an encrypted ballot $b' = (b, \check{x}, \check{w}, \check{p})$ from the ballot box, the receipt generator first checks signature and proofs (from the computer and the ballot box). If the checks are successful, it generates:

- a receipt code $\check{r} = d_V(\check{w}\check{x}^{a_3})$ sent by out-of-band channel directly to the voter. Intuitively, the receipt generator decrypts the (blinded) ballot, applying the permutation function $d_V$ associated to the voter. This gives assurance to the voter that the correct vote was submitted to the bulletin board.
- a signature on a hash of the original encrypted ballot for the ballot box. Once transmitted by the bulletin board, it allows the computer to inform the voter that his vote has been accepted.

### 2.3 Counting phase

Once the ballot box is closed, the counting phase begins (Figure 2). The ballot box selects the encrypted votes $x_1, \ldots, x_k$ which need to be decrypted (if a voter is re-voting, all the submitted ballots are in the memory of the ballot box and only the last ballot should be sent) and sends them to the decryption service. The whole content of the ballot box $b_1, \ldots, b_n$ $(n \geq k)$ is revealed to the auditor, including previous votes from re-voting voters. The receipt generator sends to the auditor the list of hashes of ballots it has seen during the submission phase. The decryption service decrypts the incoming ciphertexts $x_1, \ldots, x_k$ received from the ballot box and mix the results before outputting them $\mathsf{dec}(x_{\sigma(1)}, a_1), \ldots, \mathsf{dec}(x_{\sigma(k)}, a_1)$ where $\sigma$ denotes the permutation obtained by shuffling the votes. It also provides the auditor with a proof $pfk$ showing that the input ciphertexts and the outcoming decryption indeed match. Using the ballot box content and the list of hashes from the receipt generator, the auditor verifies that no ballots have been inserted or lost and computes his own list of encrypted ballots which should be counted. He compares this list with the one received from the decryption service and verifies the proof given by the latter.

## 3 Applied Pi Calculus

We use the framework of the applied-pi calculus [3] for formally describing the Norwegian protocol. To help with readability, the definitions of the applied-pi calculus are briefly recalled here.

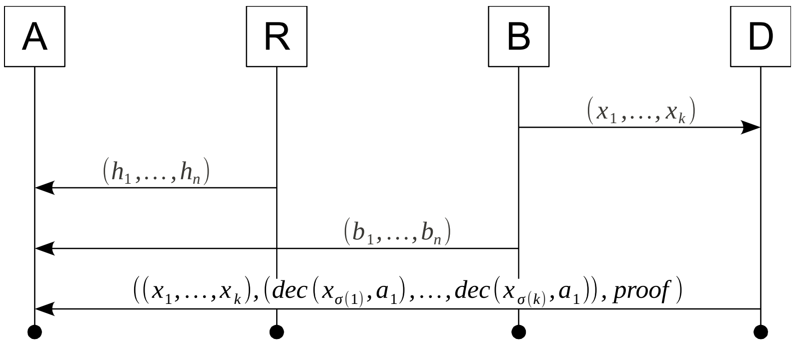

A
R
B
D
$(x_1,\dots,x_k)$
$(h_1,\dots,h_n)$
$(b_1,\dots,b_n)$
$((x_1,\dots,x_k),(dec(x_{\sigma(1)},a_1),\dots,dec(x_{\sigma(k)},a_1)), proof)$

**Fig. 2.** Counting phase.

## 3.1 Terms

As usual, messages are represented by *terms* built upon an infinite set of *names* $\mathcal{N}$ (for communication channels or atomic data), a set of *variables* $\mathcal{X}$ and a *signature* $\Sigma$ consisting of a finite set of *function symbols* (to represent cryptographic primitives). A function symbol $f$ is assumed to be given with its arity $ar(f)$. Then the set of terms $T(\Sigma,\mathcal{X},\mathcal{N})$ is formally defined by the grammar :

$$\begin{array}{ll} t, t_1, t_2, \ldots ::= & \\ \quad x & x \in \mathcal{X} \\ \quad n & n \in \mathcal{N} \\ \quad f(t_1,\ldots,t_n) & f \in \Sigma, n = ar(f) \end{array}$$

We write $\{^{M_1}/_{x_1},\ldots,{}^{M_n}/_{x_n}\}$ for the *substitution* that replaces the variables $x_i$ with the terms $M_i$. $N\sigma$ refers to the result of applying substitution $\sigma$ to the free variables of term $N$. A term is called *ground* when it does not contain variables.

In order to represent the properties of the primitives, the signature $\Sigma$ is equipped with an *equational theory* $E$ that is a set of equations which hold on terms built from the signature. We denote $=_E$ the smallest equivalence relation induced by $E$, closed under application of function symbols, substitution of terms for variables and bijective renaming of names. We write $M =_E N$ when the equation $M = N$ is in the theory $E$.

*Example 1.* A standard signature for representing encryption is $\Sigma = \{\mathsf{dec}, \mathsf{penc}\}$ where $\mathsf{penc}$ represents encryption while $\mathsf{dec}$ is decryption. Decryption is modeled by the theory $E_{\mathsf{enc}}$, defined by the equation $\mathsf{dec}(\mathsf{penc}(x,r,\mathsf{pk}(k)),k) = x$.

## 3.2 Processes

*Processes* and *extended processes* are defined in Figure 3. The process 0 represents the null process that does nothing. $P \mid Q$ denotes the parallel composition of $P$ with $Q$ while $!P$ denotes the unbounded replication of $P$ (*i.e.* the unbounded

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

**Fig. 3.** Syntax for processes

parallel composition of $P$ with itself). $\nu\, n.P$ creates a fresh name $n$ and the behaves like $P$. if $\phi$ then $P$ else $Q$ behaves like $P$ if $\phi$ holds and like $Q$ otherwise. $u(x).P$ inputs some message 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 $P$. We write $\nu\, \tilde{u}$ for the (possibly empty) series of pairwise-distinct binders $\nu\, u_1. \cdots .\nu\, u_n$. The active substitution $\{^{M}/_{x}\}$ can replace the variable $x$ for the term $M$ in every process it comes into contact with and this behaviour can be controlled by restriction, in particular, the process $\nu\, x\left(\{^{M}/_{x}\} \mid P\right)$ corresponds exactly to let $x = M$ in $P$. As in [10], we slightly extend the applied-pi calculus by letting conditional branches now depend on formulae $\phi, \psi ::= M = N \mid M \neq N \mid \phi \wedge \psi$. If $M$ and $N$ are ground, we define $[\![M = N]\!]$ to be true if $M =_E N$ and false otherwise. The semantics of $[\![\ ]\!]$ is then extended to formulae as expected.

The *scope* of names and variables are delimited by binders $u(x)$ and $\nu\,(u)$. Sets of bound names, bound variables, free names and free variables are respectively written $\mathrm{bn}(A)$, $\mathrm{bv}(A)$, $\mathrm{fn}(A)$ and $\mathrm{fv}(A)$. Occasionally, we write $\mathrm{fn}(M)$ (respectively $\mathrm{fv}(M)$) 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 instead of an extended process. We obtain $C[A]$ as the result of filling $C[\_]$'s hole with the extended process $A$. An *evaluation context* is a context whose hole is not in the scope of a replication, a conditional, an input or an output. A context $C[\_]$ closes $A$ when $C[A]$ is closed.

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 $\varphi$, denoted $\mathrm{dom}(\varphi)$ is the set of variables for which $\varphi$ 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.

| | | |
|---|---|---|
| Par-0 | $A \equiv A \mid 0$ | |
| Par-A | $A \mid (B \mid C) \equiv (A \mid B) \mid C$ | |
| Par-C | $A \mid B \equiv B \mid A$ | |
| Repl | $!P \equiv P \mid !P$ | |
| New-0 | $\nu n.0 \equiv 0$ | |
| New-C | $\nu u.\nu w.A \equiv \nu w.\nu u.A$ | |
| New-Par | $A \mid \nu u.B \equiv \nu u.(A \mid B)$ | if $u \notin \mathrm{fv}(A) \cup \mathrm{fn}(A)$ |
| Alias | $\nu x.\{^M/_x\} \equiv 0$ | |
| Subst | $\{^M/_x\} \mid A \equiv \{^M/_x\} \mid A\{^M/_x\}$ | |
| Rewrite | $\{^M/_x\} \equiv \{^N/_x\}$ | if $M =_E N$ |

**Fig. 4.** Structural equivalence.

## 3.3 Operational 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* ($\xrightarrow{\alpha}$). *Structural equivalence* is defined in Figure 4. It is closed by $\alpha$-conversion of both bound names and bound variables, and closed under application of evaluation contexts. The *internal reductions* and *labelled reductions* are defined in Figure 5. They are closed under structural equivalence and application of evaluation contexts. Internal reductions represent evaluation of condition and internal communication between processes. Labelled reductions represent communications with the environment.

## 3.4 Equivalences

Privacy properties are often stated as equivalence relations [12]. Intuitively, if a protocol preserves ballot secrecy, an attacker should not be able to distinguish between a scenario where a voter votes 0 from a scenario where the voter votes 1. *Static equivalence* formally expresses indistinguishability of sequences of terms.

**Definition 1 (Static equivalence).** *Two closed frames $\varphi$ and $\psi$ are statically equivalent, denoted $\varphi \approx_s \psi$, if $dom(\varphi) = dom(\psi)$ and there exists a set of names $\tilde{n}$ and substitutions $\sigma, \tau$ such that $\varphi \equiv \nu \tilde{n}.\sigma$ and $\psi \equiv \nu \tilde{n}.\tau$ and for all terms $M, N$ such that $\tilde{n} \cap (fn(M) \cup fn(N)) = \emptyset$, we have $M\sigma =_E N\sigma$ holds if and only if $M\tau =_E N\tau$ holds. Two closed extended processes $A, B$ are statically equivalent, written $A \approx_s B$, if their frames are statically equivalent; that is, $\varphi(A) \approx_s \varphi(B)$.*

*Example 2.* Consider the signature and equational theory $E_{\mathsf{enc}}$ defined in Example 1. Let $\varphi_1 = \nu k.\sigma_1$ and $\varphi_2 = \nu k.\sigma_2$ where $\sigma_1 = \{^{\mathsf{penc}(s_1,r_1,\mathsf{pk}(k))}/_{x_1}, {}^{\mathsf{pk}(k)}/_{x_2}\}$, $\sigma_2 = \{^{\mathsf{penc}(s_2,r_2,\mathsf{pk}(k))}/_{x_1}, {}^{\mathsf{pk}(k)}/_{x_2}\}$ and $s_1$, $s_2$, $k$ are names. We have that $\varphi_1 \not\approx_s \varphi_2$. Indeed, we $\mathsf{penc}(s_1, r_1, x_2)\sigma_1 =_E x_1\sigma_1$ but $\mathsf{penc}(s_1, r_1, x_2)\sigma_2 \neq_E x_1\sigma_2$. However, we have that $\nu k, r_1.\sigma_1 \approx_s \nu k, r_2.\sigma_2$.

$$
\begin{array}{ll}
(\textsc{Comm}) & \overline{c}\langle x\rangle.P \mid c(x).Q \to P \mid Q \\
(\textsc{Then}) & \text{if } \phi \text{ then } P \text{ else } Q \to P \ \text{ if } [\![\phi]\!] = \mathsf{true} \\
(\textsc{Else}) & \text{if } \phi \text{ then } P \text{ else } Q \to Q \ \text{ otherwise}
\end{array}
$$

$$
\begin{array}{ll}
(\textsc{In}) & c(x).P \xrightarrow{c(M)} P\{^{M}/_{x}\} \\[1ex]
(\textsc{Out-Atom}) & \overline{c}\langle u\rangle.P \xrightarrow{\overline{c}\langle u\rangle} P \\[1ex]
(\textsc{Open-Atom}) & \dfrac{A \xrightarrow{\overline{c}\langle u\rangle} A' \qquad u \neq c}{\nu\, u.A \xrightarrow{\nu\, u.\overline{c}\langle u\rangle} A'} \\[2ex]
(\textsc{Scope}) & \dfrac{A \xrightarrow{\alpha} A' \qquad u \text{ does not occur in } \alpha}{\nu\, u.A \xrightarrow{\alpha} \nu\, u.A'} \\[2ex]
(\textsc{Par}) & \dfrac{A \xrightarrow{\alpha} A' \qquad \mathrm{bv}(\alpha) \cap \mathrm{fv}(B) = \mathrm{bn}(\alpha) \cap \mathrm{fn}(B) = \emptyset}{A \mid B \xrightarrow{\alpha} A' \mid B} \\[2ex]
(\textsc{Struct}) & \dfrac{A \equiv B \qquad B \xrightarrow{\alpha} B' \qquad B' \equiv A'}{A \xrightarrow{\alpha} A'}
\end{array}
$$

where $\alpha$ is a *label* of the form $c(M)$, $\overline{c}\langle u\rangle$, or $\nu\, u.\overline{c}\langle u\rangle$ such that $u$ is either a channel name or a variable of base type.

**Fig. 5.** Semantics for processes

Observational equivalence is the active counterpart of static equivalence, where the attacker can actively interact with the processes. The definition of observational equivalence requires to reason about all contexts (*i.e.* all adversaries), which renders the proofs difficult. Since observational equivalence has been shown to coincide [3,22] with labelled bisimilarity, we adopt the later in this paper.

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

Examples of labelled bisimilar processes will be provided in Section 5.

## 4 Modelling the protocol in applied-pi calculus

We now provide a formal specification of the protocol, using the framework of the applied-pi calculus, defined in the previous section. The first step consists in modeling the cryptographic primitives used by the protocol.

## 4.1 Equational theory

We adopt the following signature to capture the cryptographic primitives used by the protocol.

$$\Sigma_{sign} = \{\mathsf{Ok}, \mathsf{fst}, \mathsf{hash}, \mathsf{p}, \mathsf{pk}, \mathsf{s}, \mathsf{snd}, \mathsf{vk}, \mathsf{blind}, \mathsf{d}, \mathsf{dec}, +, *, \circ, \diamond, \mathsf{pair},$$
$$\mathsf{renc}, \mathsf{sign}, \mathsf{unblind}, \mathsf{checkpfk}_1, \mathsf{checkpfk}_2, \mathsf{checksign}, \mathsf{penc}, \mathsf{pfk}_1, \mathsf{pfk}_2\}$$

with function $\mathsf{Ok}$ is a constant ; $\mathsf{fst}$, $\mathsf{hash}$, $\mathsf{p}$, $\mathsf{pk}$, $\mathsf{s}$, $\mathsf{snd}$, $\mathsf{vk}$ are unary functions ; $\mathsf{blind}$, $\mathsf{d}$, $\mathsf{dec}$, $+$, $*$, $\circ$, $\diamond$, $\mathsf{pair}$, $\mathsf{renc}$, $\mathsf{sign}$, $\mathsf{unblind}$ are binary functions ; $\mathsf{checkpfk}_1$, $\mathsf{checkpfk}_2$, $\mathsf{checksign}$, $\mathsf{penc}$ are ternary functions and $\mathsf{pfk}_1$, $\mathsf{pfk}_2$ are quaternary functions.

The term $\mathsf{pk}(K)$ denotes the public key corresponding to the secret key $K$ in asymmetric encryption. Terms $\mathsf{s}(I)$, $\mathsf{p}(I)$, and $\mathsf{vk}(I)$ are respectively the blinding factor, the parameter and the verification key associated to a secret id $I$. The specific coding function used by the receipt generator for a voter with secret id $I$, applied to a message $M$ is represented by $\mathsf{d}(\mathsf{p}(I), M)$. It corresponds to the function $d_I(M)$ explained in Section 2.2. The term $\mathsf{blind}(M, N)$ the message $M$ blinded by $N$. Unblinded such a blinded term $P$, using the same blinding factor $N$ is denoted by $\mathsf{unblind}(P, N)$. The term $\mathsf{penc}(M, N, P)$ refers to the encryption of plaintext $M$ using random nonce $N$ and public key $P$. The term $M \circ N$ denotes the homomorphic combination of ciphertexts $M$ and $M'$ and the corresponding operation on plaintexts is written $P \diamond Q$ and $R * S$ on nonces. The decryption of ciphertext $C$ using secret key $K$ is denoted $\mathsf{dec}(C, K)$. The term $\mathsf{renc}(M, K)$ is the re-encryption of the ciphertext $M$ using a secret key $K$ and leads to another ciphertext of the same plaintext with the same nonce but a different public key. The operation between secret keys is denoted by $K + L$. The term $\mathsf{sign}(M, N)$ refers to the signature of the message $M$ using secret id $N$. The term $\mathsf{pfk}_1(M, N, P, Q)$ represents a proof of knowledge that proves that $Q$ is a ciphertext on the plaintext $P$ using nonce $N$. The term $\mathsf{pfk}_2(M, N, P, Q)$ denotes another proof of knowledge proving that $Q$ is either a re-encryption or a masking of a term $P$ using a secret key or nonce $N$. We introduce tuples using pairings and, for convenience, $\mathsf{pair}(M_1, \mathsf{pair}(\ldots, \mathsf{pair}(M_{n-1}, M_n)))$ is abbreviated as $(M_1, \ldots, M_n)$ and $\mathsf{fst}(\mathsf{snd}^{i-1}(M))$ is denoted $\Pi_i$ with $i \in \mathbb{N}$.

The properties of the primitives are then modelled by equipping the signature with an equational theory $E$ that asserts functions $+$, $*$, $\circ$ and $\diamond$ are commutative and associative, and includes the equations defined in Figure 6. The three first equations are quite standard. Equation (4) allows to decrypt a blinded ciphertext in order to get the corresponding blinded plaintext. Equation (5) models the homomorphic combination of ciphertexts. Equation (6) represents the re-encryption of a ciphertext. The operation of unblinding is described through Equation (7). Equations (8), (9) and (10) allows respectively the verification of signatures and proofs of knowledge for $\mathsf{pfk}_1$ and $\mathsf{pfk}_2$ proofs.

## 4.2 Norwegian protocol process specification

The description of the processes representing the actors of the protocol makes use of auxiliary checks that are defined in Figure 7. We did not model re-voting

$$\mathsf{fst}(\mathsf{pair}(x,y)) = x \tag{1}$$
$$\mathsf{snd}(\mathsf{pair}(x,y)) = y \tag{2}$$
$$\mathsf{dec}(\mathsf{penc}(x_{plain}, x_{rand}, \mathsf{pk}(x_{sk})), x_{sk}) = x_{plain} \tag{3}$$
$$\mathsf{dec}(\mathsf{blind}(\mathsf{penc}(x_{plain}, x_{rand}, \mathsf{pk}(x_{sk})), x_{blind}), x_{sk}) = \mathsf{blind}(x_{plain}, x_{blind}) \tag{4}$$
$$\begin{aligned}\mathsf{penc}(x_{pl}, x_{rand}, x_{pub}) \circ \mathsf{penc}(y_{pl}, y_{rand}, x_{pub}) =& \\ &\mathsf{penc}(x_{pl} \diamond y_{pl}, x_{rand} * y_{rand}, x_{pub})\end{aligned} \tag{5}$$
$$\begin{aligned}\mathsf{renc}(\mathsf{penc}(x_{plain}, x_{rand}, \mathsf{pk}(x_{sk})), y_{sk}) =& \\ &\mathsf{penc}(x_{plain}, x_{rand}, \mathsf{pk}(x_{sk} + y_{sk}))\end{aligned} \tag{6}$$
$$\mathsf{unblind}(\mathsf{blind}(x_{plain}, x_{blind}), x_{blind}) = x_{plain} \tag{7}$$
$$\mathsf{checksign}(x_{plain}, \mathsf{vk}(x_{id}), \mathsf{sign}(x_{plain}, x_{id})) = \mathsf{Ok} \tag{8}$$
$$\begin{aligned}\mathsf{checkpfk}_1(\mathsf{vk}(x_{id}), \mathsf{ball}, \mathsf{pfk}_1(x_{id}, x_{rand}, x_{plain}, \mathsf{ball})) = \mathsf{Ok}& \\ \text{where } \mathsf{ball} = \mathsf{penc}(x_{plain}, x_{rand}, x_{pub})&\end{aligned} \tag{9}$$
$$\begin{aligned}\mathsf{checkpfk}_2(\mathsf{vk}(x_{id}), \mathsf{ball}, \mathsf{pfk}_2(\mathsf{vk}(x_{id}), x_{bk}, x_{plain}, \mathsf{ball})) = \mathsf{Ok}& \\ \text{where } \mathsf{ball} = \mathsf{renc}(x_{plain}, x_{bk}) \text{ or } \mathsf{ball} = \mathsf{blind}(x_{plain}, x_{bk})&\end{aligned} \tag{10}$$

**Fig. 6.** Equations for encryption, blinding, signature and proof of knowledge.

since it is explicitely and strongly discouraged in [15], as it may allow an attacker to swap two votes (the initial casted one and its revoted one).

The voting process $V$ represents both the voter and his computer. It is parametrized by a free variable $x_{vote}$ representing voter's vote and free names $c_{auth}$, $c_{RV}$ which denote the channel shared with the voter and, respectively, the ballot box and the receipt generator. $g_1$ is a variable representing the public key of the election, $id$ is the secret id of the voter and $idp_R$ is a variable representing the verification key of the receipt generator. Note that messages sent over $c_{auth}$ and $c_{RV}$ are also sent on the public channel $c_{out}$ to the adversary, to simulate authenticated but not confidential channels.

$$\begin{aligned}\phi_{\mathsf{b}}(idp_i, x) = [&(\Pi_1(x), \Pi_2(x), \Pi_3(x)) = x \\ &\wedge \mathsf{checksign}((\Pi_1(x), \Pi_2(x)), \mathsf{vk}(id_i), \Pi_3(x)) = \mathsf{Ok} \\ &\wedge\ \mathsf{checkpfk}_1(idp_i, \Pi_1(x), \Pi_2(x)) = \mathsf{Ok}]\end{aligned}$$

$$\phi_{\mathsf{s}}(idp_R, x, y) = [\mathsf{checksign}(x, idp_R, y) = \mathsf{Ok}]$$

$$\phi_{\mathsf{v}}(idp_R, id_i, x, y, v, z) = [\mathsf{checksign}(x, idp_R, y) = \mathsf{Ok}\ \wedge\ \mathsf{d}(\mathsf{p}(id_i), \mathsf{blind}(v, \mathsf{s}(id_i))) = z]$$

$$\begin{aligned}&(\forall k = 1..3,\ \ x_i^k = \Pi_k(\Pi_1(x)),\ \forall k = 4..7,\ \ x_i^k = \Pi_{k-2}(x)) \\ &\phi_{\mathsf{r}}(idp_i, x) = [(x_i^1, x_i^2, x_i^3) = \Pi_1(x) \wedge (\Pi_1(x), x_i^4, x_i^5, x_i^6, x_i^7) = x \\ &\qquad \wedge\ \mathsf{checksign}((x_i^1, x_i^2), idp_i, x_i^3) = \mathsf{Ok} \wedge \mathsf{checkpfk}_1(idp_i, x_i^1, x_i^2) = \mathsf{Ok} \\ &\qquad \wedge\ \mathsf{checkpfk}_2(idp_i, x_i^4, x_i^5) = \mathsf{Ok} \wedge \mathsf{checkpfk}_2(idp_i, x_i^6, x_i^7) = \mathsf{Ok}]\end{aligned}$$

**Fig. 7.** Auxiliary checks performed by the participants to the protocol.

```
$V(c_{auth}, c_{out}, c_{RV}, g_1, id, idp_R, x_{vote}) = \nu\, t$ .
    let $e = \mathsf{penc}(x_{vote}, t, g_1)$ in
    let $p = \mathsf{pfk}_1(id, t, x_{vote}, e)$ in
    let $si = \mathsf{sign}((e, p), id)$ in
    $\overline{c_{out}}\langle (e, p, si)\rangle$ .
    $\overline{c_{auth}}\langle (e, p, si)\rangle$ .                      % encrypted ballot sent to B
    $c_{RV}(x)$ . $c_{auth}(y)$ .
    $\overline{c_{out}}\langle x\rangle$ . $\overline{c_{out}}\langle y\rangle$ .
    let $hv = \mathsf{hash}((\mathsf{vk}(id), e, p, si))$ in          % recomputes what should
                                                       be sent by R
    if $\phi_{\mathsf{v}}(idp_R, id, h, x, x_{vote}, y)$ then $\overline{c_{auth}}\langle \mathsf{Ok}\rangle$    % checks validity
```

Process $B_n$ corresponds to the ballot box, ready to listen to $n$ voters. The ballots are coming from authenticated channels $c_1, \ldots, c_n$ and the ballot box can send messages to the receipt generator, the decryption service and the auditor through secure channels $c_{BR}$, $c_{BD}$ and $c_{BA}$. The parameters of the ballot box are keys : $g_1$, $g_3$ (public) and $a_2$ (secret); public ids of voters $idp_1, \ldots, idp_n$ (*i.e.* verification keys) and corresponding blinding factors $s_1, \ldots, s_n$. (Step $c(sy_1)$ is a technical synchronisation, it does not appear in the real specification.)

```
$B_n(c_{BR}, c_{BD}, g_1, a_2, g_3, idp_R, c_1, idp_1, s_1, \ldots, c_n, idp_n, s_n) =$
    $\ldots$ . $c_i(x_i)$ .
    if $\phi_{\mathsf{b}}(idp_i, x_i)$ then                      % checks validity of ballot
    let $e_i = \mathsf{renc}(\Pi_1(x_i), a_2)$ in
    let $pfk_i^e = \mathsf{pfk}_2(idp_i, a_2, \Pi_1(x_i), e_i)$ in
    let $b_i = \mathsf{blind}(e_i, s_i)$ in
    let $pfk_i^b = \mathsf{pfk}_2(idp_i, s_i, e_i, b_i)$ in         % computes re-encrypted masked
                                                  ballot and corresponding proofs.
    $\overline{c_{BR}}\langle (x_i, e_i, pfk_i^e, b_i, pfk_i^b)\rangle . c_{BR}(y_i)$.   % message sent to R
    let $hb_i = \mathsf{hash}((\mathsf{vk}(id_i), \Pi_1(x_i), \Pi_2(x_i), \Pi_3(x_i)))$ in
    if $\phi_{\mathsf{s}}(idp_R, hb_i, y_i)$ then                 % checks validity of confirmation
    $\overline{c_i}\langle y_i\rangle$ . $c_i(sy_i)$ $\ldots$                    % transmit confirmation to the voter
    $\overline{c_n}\langle y_n\rangle$ . $c_n(sy_n)$ .
    $\overline{c_{BD}}\langle \Pi_1(x_1)\rangle$ . $\ldots$ . $\overline{c_{BD}}\langle \Pi_1(x_n)\rangle$ .   % output encrypted votes to the
                                                    Decryption Service
    $\overline{c_{BA}}\langle x_1\rangle$ . $\ldots$ . $\overline{c_{BA}}\langle x_n\rangle$             % output the content to the Auditor
```

Receipt generator's process is denoted by $R_n$. It deals with the ballot box and the auditor through secure channels $c_{BR}$ and $c_{RA}$ and directly with voters through out-of-band channels $c_{RV_1}, \ldots, c_{RV_n}$. It is parametrized with keys: $g_1$, $g_2$ (public) and $a_3$ (secret); the public ids of voters and corresponding receipt coding functions parametrized by $pr_1, \ldots, pr_n$.

```
$R_n(c_{BR}, g_1, g_2, a_3, id_R, c_{RV_1}, idp_1, pr_1, \ldots, c_{RV_n}, idp_n, pr_n) =$
    $\ldots$ . $c_{BR}(x_i)$ .
    let $x_i^k = \Pi_k(\Pi_1(x_i)),\ k = 1..3$ in
    let $x_i^k = \Pi_{k-2}(x_i),\ k = 4...7$ in
    if $\phi_\mathsf{r}(idp_i, x_i)$ then                 % checks ballot box's computations
    let $hbr_i = \mathsf{hash}((idp_i, x_i^1, x_i^2, x_i^3))$ in
    let $hbpr_i = \mathsf{hash}((idp_i, x_i^1, x_i^2))$ in
    let $r_i = \mathsf{d}(pr_i, \mathsf{dec}(x_i^6, a_3))$ in          % computes the receipt code for V
    let $sig_i = \mathsf{sign}(hbr_i, id_R)$ in            % computes confirmation for B
    $\overline{c_{RV_i}}\langle r_i\rangle\ .\ \overline{c_{BR}}\langle sig_i\rangle\ .\ \ldots$
    $\overline{c_{RV_n}}\langle r_n\rangle\ .\ \overline{c_{BR}}\langle sig_n\rangle\ .\ \ldots$
    $\overline{c_{RA}}\langle(idp_1, hbpr_1, hbr_1)\rangle\ .\ \ldots\ .\ \overline{c_{RA}}\langle(idp_n, hbpr_n, hbr_n)\rangle$
                                                   % output content to the Auditor
```

The decryption service is represented by process $D_n$. Communicating securely with the ballot box and the auditor through channels $c_{BD}$ and $c_{DA}$, it also outputs results through public channel $c_{out}$. In order to decrypt ballots, it needs to know the secret key $a_1$. We model two processes, one including a swap between the two first votes, to model the shuffling which is necessary to ensure ballot secrecy.

```
$D_n(c_{BD}, c_{DA}, c_{out}, a_1) =$
    $c_{BD}(x_1)\ .\ \ldots\ .\ c_{BD}(x_n)\ .$
    $\overline{c_{DA}}\langle\mathsf{hash}((x_1, \ldots, x_n))\rangle\ .\ c_{DA}(x)\ .$   % creating hash of ciphertexts and
                                                 waiting for auditor's approval
    let $dec_k = \mathsf{dec}(x_k, a_1),\ k = 1..n$ in   % decryption of ciphertexts
    $\overline{c_{out}}\langle dec_1\rangle\ .\ \ldots\ .\ \overline{c_{out}}\langle dec_n\rangle$          % publication of results
```

```
$\overline{D}_n(c_{BD}, c_{DA}, c_{out}, a_1) =$
    $c_{BD}(x_1)\ .\ \ldots\ .\ c_{BD}(x_n)\ .$
    $\overline{c_{DA}}\langle\mathsf{hash}((x_1, \ldots, x_n))\rangle\ .\ c_{DA}(x)\ .$
    let $dec_1 = \mathsf{dec}(x_2, a_1)$ in              % the swap between the two first
    let $dec_2 = \mathsf{dec}(x_1, a_1)$ in                votes is modelled here
    let $dec_k = \mathsf{dec}(x_k, a_1),\ k = 3..n$ in
    $\overline{c_{out}}\langle dec_1\rangle\ .\ \ldots\ .\ \overline{c_{out}}\langle dec_n\rangle$
```

Finally, the auditor process, $AD_n$, communicates with the other infrastructure players using secure channels $c_{BA}$, $c_{RA}$ and $c_{DA}$. It knows public ids of voters.

```
$AD_n(c_{BA}, c_{RA}, c_{DA}, idp_1, \ldots, idp_n) =$
    $c_{DA}(h_d)\ .$                              % input of contents of B, R and D
    $c_{BA}(x_1)\ .\ \ldots\ .\ c_{BA}(x_n)\ .\ c_{RA}(h_1)\ .\ \ldots\ .\ c_{RA}(h_1)\ .$
    let $hba_i = \mathsf{hash}((\Pi_1(x_i), \Pi_2(x_i), \Pi_3(x_i)))$ in
    let $hbpa_i = \mathsf{hash}((\Pi_1(x_i), \Pi_2(x_i)))$ in
    let $ha = \mathsf{hash}((\Pi_1(x_1), \ldots, \Pi_n(x_n)))$ in
    if $\phi_\mathsf{a}(x_1, h_1, idp_1, \ldots, x_n, h_n, idp_n, h, h_d)$ then $\overline{c_{DA}}\langle\mathsf{Ok}\rangle$ else 0
                                              % checks and approval sent to D.
```

where $\phi_\mathsf{a}(x_1, h_1, idp_1, \ldots, x_n, h_n, idp_n, h, h_d) = [(\Pi_1(x_i), \Pi_2(x_i), \Pi_3(x_i)) = x_i$

$\wedge(\Pi_1(h_i), \Pi_2(h_i), \Pi_3(h_i)) = h_i \wedge \ \Pi_2(h_i) = hbp_i \wedge \Pi_3(h_i) = hb_i \wedge h_d = h$
$\wedge\ \mathsf{checksign}((\Pi_1(x_i), \Pi_2(x_i)), idp_i, \Pi_3(x_i)) = \mathsf{Ok}]$

The interaction of all the players is simply modeled by considering all the processes in parallel, with the correct instantiation and restriction of the parameters. In what follows, the restricted name $a_1$, $a_2$, $a_3$ model secret keys used in the protocol and public keys $\mathsf{pk}(a_1)$, $\mathsf{pk}(a_2)$ and $\mathsf{pk}(a_3)$ are added in the process frame. The restricted name $c_1$, $c_2$ and $c_{RV_1}$, $c_{RV_2}$ model authentic channels between honest voters and, respectively, the ballot box and the receipt generator. The restricted name $id_1$, $id_2$, $id_R$ represent secret ids of honest voters and receipt generator, the corresponding public id's are added in the process's frame.

Then the setting of the authorities is modeled by $A_n\,[\_]$ where $n$ is the number of voters and the hole is the voter place. $A_n\,[\_]$ is the analogue of $\overline{A}_n\,[\_]$ with the Decryption service swapping the two first votes (its use will be clearer in the next section, when defining vote privacy).

$$\begin{aligned}
\tilde{n} &= (a_1, a_2, id_1, id_2, id_R, c_1, c_2, c_{RV_1}, c_{RV_2}, c_{BR}, c_{BD}, c_{BA}, c_{RA}, c_{DA}) \\
\Gamma &= \{^{\mathsf{pk}(a_1)}/_{g_1}, {}^{\mathsf{pk}(a_2)}/_{g_2}, {}^{\mathsf{pk}(a_3)}/_{g_3}, {}^{\mathsf{vk}(id_1)}/_{idp_1}, \ldots, {}^{\mathsf{vk}(id_n)}/_{idp_n}, {}^{\mathsf{vk}(id_R)}/_{idp_R}\} \\
A_n\,[\_] &= \nu\,\tilde{n}\ .(\text{let } a_3 = a_1 + a_2 \text{ in } [\_|B_n\{^{\mathsf{s}(id_1)}/_{s_1}, \cdots, {}^{\mathsf{s}(id_n)}/_{s_n}\} \\
&\qquad |R_n\{^{\mathsf{p}(id_1)}/_{pr_1}, \cdots, {}^{\mathsf{p}(id_n)}/_{pr_n}\}|D_n|AD_n|\Gamma]) \\
\overline{A}_n\,[\_] &= \nu\,\tilde{n}\ .(\text{let } a_3 = a_1 + a_2 \text{ in } [\_|B_n\{^{\mathsf{s}(id_1)}/_{s_1}, \cdots, {}^{\mathsf{s}(id_n)}/_{s_n}\} \\
&\qquad |R_n\{^{\mathsf{p}(id_1)}/_{pr_1}, \cdots, {}^{\mathsf{p}(id_n)}/_{pr_n}\}|\overline{D}_n|AD_n|\Gamma])
\end{aligned}$$

The frame $\Gamma$ represents the initial knowledge of the attacker: it has access to the public keys of the authorities and the verification keys of the voters. Moreover, since only the two first voters are assumed to be honest, only their two secret ids are restricted (in $\tilde{n}$). The attacker has therefore access to the secret ids of all the other voters. Parameters of subprocesses are left implicit except for $s_1, \ldots, s_n$ for the ballot box and $pr_1, \ldots, pr_n$ for the receipt generator which are respectively replaced by $\mathsf{s}(id_1), \ldots, \mathsf{s}(id_n)$, the blinding factors, and $\mathsf{p}(id_1), \ldots, \mathsf{p}(id_n)$, used to distinguish the coding dunction associated to a voter.

## 5 Formal analysis of ballot secrecy

Our analysis shows that the Norwegian e-voting protocol preserves ballot secrecy, even when all but two voters are corrupted, provided that the other components are honest. We also identified several cases of corruption that are subject to attacks. Though not surprising, these cases were not previously mentioned in the literature.

### 5.1 Ballot secrecy with corrupted voters

Ballot secrecy has been formalized in terms of equivalence by Delaune, Kremer, and Ryan in [12]. 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:

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

We are able to show that the Norwegian protocol preserves ballot secrecy, even all but two voters are corrupted.

**Theorem 1.** *Let $n$ be the number of voters. The Norwegian e-voting protocol process specification satisfies ballot secrecy with the auditing process, even with $n-2$ voters are corrupted, provided that the other components are honest.*

$$A_n[V\{^{c_1}/_{c_{auth}}, ^{c_{RV_1}}/_{c_{RV}}\}\sigma \mid V\{^{c_2}/_{c_{auth}}, ^{c_{RV_2}}/_{c_{RV}}\}\tau] \\ \approx_l \overline{A}_n\,[V\{^{c_1}/_{c_{auth}}, ^{c_{RV_1}}/_{c_{RV}}\}\tau | V\{^{c_2}/_{c_{auth}}, ^{c_{RV_2}}/_{c_{RV}}\}\sigma]$$

*where* $\sigma = \{^{v_1}/_{x_{vote}}\}$ *and* $\tau = \{^{v_2}/_{x_{vote}}\}$.

We can also show ballot secrecy, without an auditor. This means that the auditor does not contribute to ballot secrecy in case the administrative components are honest (which was expected). Formally, we define $A'_n\,[\_]$ and $\overline{A}_n\,[\_]'$ to be the analog of $A_n\,[\_]$ and $\overline{A}_n\,[\_]$, removing the auditor.

**Theorem 2.** *Let $n$ be the number of voters. The Norwegian e-voting protocol process specification satisfies ballot secrecy without the auditing process, even with $n-2$ voters are corrupted, provided that the other components are honest.*

$$A'_n[V\{^{c_1}/_{c_{auth}}, ^{c_{RV_1}}/_{c_{RV}}\}\sigma \mid V\{^{c_2}/_{c_{auth}}, ^{c_{RV_2}}/_{c_{RV}}\}\tau] \\ \approx_l \overline{A'}_n\,[V\{^{c_1}/_{c_{auth}}, ^{c_{RV_1}}/_{c_{RV}}\}\tau | V\{^{c_2}/_{c_{auth}}, ^{c_{RV_2}}/_{c_{RV}}\}\sigma]$$

*where* $\sigma = \{^{v_1}/_{x_{vote}}\}$ *and* $\tau = \{^{v_2}/_{x_{vote}}\}$.

The proof of Theorems 1 and 2 works in two main steps. First we guess a relation $\mathcal{R}$ such that for any two processes $P, Q$ in relation $(P\mathcal{R}Q)$ any move of $P$ can be matched by a move of $Q$ such that the resulting processes remain in relation. This amounts to characterize all possible successors of $A_n[V\{^{c_1}/_{c_{auth}}, ^{c_{RV_1}}/_{c_{RV}}\}\sigma \mid V\{^{c_2}/_{c_{auth}}, ^{c_{RV_2}}/_{c_{RV}}\}\tau]$ and $\overline{A}_n[V\{^{c_1}/_{c_{auth}}, ^{c_{RV_1}}/_{c_{RV}}\}\tau | V\{^{c_2}/_{c_{auth}}, ^{c_{RV_2}}/_{c_{RV}}\}\sigma]$. We show in particular that whenever the attacker sends a term $N$ that is accepted by the ballot box for a voter with secret id $\mathsf{id}$, then $N$ is necessarily an $\mathsf{id}$ - *valid ballot* for the following definition.

**Definition 3.** *Let $id \in \{id_1, \ldots, id_n\}$. A term $N$ is said to be a* $\mathsf{id}$ - valid ballot *if $\phi_{\mathsf{b}}(id, N) = \mathsf{true}$, equivalently :*

$$\begin{cases} N = (N_1, N_2, N_3) \\ \mathsf{checksign}((N_1, N_2), \mathsf{vk}(id), N_3) =_E \mathsf{Ok} \\ \mathsf{checkpfk}_1(\mathsf{vk}(id), N_1, N_2) =_E \mathsf{Ok} \end{cases}.$$

The second and most involved step of the proof consists in showing that the sequences of messages observed by the attacker remain in static equivalence. This requires to prove an infinite number of static equivalences. Let us introduce some notations.

$$\begin{aligned}
\theta_{sub} = {} & \{^{\mathsf{pk}(a_1)}/_{g_1}\}|\{^{\mathsf{pk}(a_2)}/_{g_2}\}|\{^{\mathsf{pk}(a_3)}/_{g_3}\}|\{^{\mathsf{vk}(id_R)}/_{idp_R}\}|\{^{ball_1}/_{b_1}\}|\{^{ballo_2}/_{b_2}\}| \\
& \{\{^{\mathsf{vk}(id_i)}/_{idp_i}\}|\ i = 1..n\}|\{\{^{\mathsf{d}(\mathsf{p}(id_i),\mathsf{dec}(\mathsf{blind}(\mathsf{renc}(\Pi_1(x_i),a_2),\mathsf{s}(id_i)),a_3))}/_{y_i}\}| \\
& \{^{\mathsf{sign}(\mathsf{hash}((\mathsf{vk}(id_i),x_i)),id_R)}/_{z_i}\}|\ i = 1..n\} \\
\Sigma_L = {} & \{^{v_1}/_{x^1_{vote}},^{v_2}/_{x^2_{vote}}\} \\
\Sigma_R = {} & \{^{v_2}/_{x^1_{vote}},^{v_1}/_{x^2_{vote}}\} \\
\theta_{ct} = {} & \{^{\mathsf{dec}(\Pi_1(x_1),a_1)}/_{result_1},^{\mathsf{dec}(\Pi_1(x_2),a_1)}/_{result_2},^{\mathsf{dec}(\Pi_1(x_i),a_1)}/_{result_i}|i = 3..n\} \\
\overline{\theta}_{ct} = {} & \{^{\mathsf{dec}(\Pi_1(x_2),a_1)}/_{result_1},^{\mathsf{dec}(\Pi_1(x_1),a_1)}/_{result_2},^{\mathsf{dec}(\Pi_1(x_i),a_1)}/_{result_i}|i = 3..n\}
\end{aligned}$$

where $ball_1$ and $ball_2$ are the terms sent by the two honest voters.

The frame $\theta_{sub}$ represents the messages sent over the (public) network during the submission phase. $\Sigma_L$ represents the scenario where voter 1 votes $v_1$ and voter 2 votes $v_2$ while $\Sigma_L$ represents the opposite scenario. $\theta_{ct}$ and $\overline{\theta}_{ct}$ represent the results published by the decryption service.

All voters with secret id $\mathsf{id}_i$ with $i \geq 3$ are corrupted. Therefore, the attacker can submit any deducible term as a ballot, that is any term that can be represented by $N_i$ with $\mathrm{fv}(N_i) \subseteq \mathrm{dom}(\theta_{sub})\backslash\{y_j, z_j\}_{j\geq i}$ (*i.e.* a recipe that can only re-use previously received messages). We are able to show that whenever the message submitted by the attacker is accepted by the ballot box, then $N_i\theta_{sub}\Sigma$ is necessarily an id$_i$-valid ballots for $\Sigma \in \{\Sigma_L, \Sigma_R\}$.

A key result of our proof is that the final frames are in static equivalence, for any behavior of the corrupted users (reflected in the $N_i$).

**Proposition 1.** *Let $N_i\theta_{sub}\Sigma$ be id$_i$-valid ballots for $\Sigma \in \{\Sigma_L, \Sigma_R\}$ and $i \in \{3, \ldots, n\}$, we have: $\nu\tilde{n}.(\theta_{sub}|\theta_{ct})\sigma_{\tilde{N}}\Sigma_L \approx_s \nu\tilde{n}.(\theta_{sub}|\overline{\theta}_{ct})\sigma_{\tilde{N}}\Sigma_R$, where $\sigma_{\tilde{N}} = \{^{ball_1}/_{x_1},\ ^{ball_2}/_{x_2},\ ^{N_j}/_{x_j}|\ j \in \{3, \ldots, n\}\}$.*

## 5.2 Attacks

Our two previous results of ballot secrecy hold provided all the administrative components (bulletin box, receipt generator, decryption service, and auditor) behave honestly. However, in order to enforce the level of trust, the voting system should remain secure even if some administrative components are corrupted. We describe two cases of corruption where ballot secrecy is no longer guaranteed.

*Dishonest decryption service.* The decryption service is a very sensitive component since it has access to the decryption key $a_1$ of the public key used for the election. Therefore, a corrupted decryption service can very easily decrypt all encrypted ballots and thus learns the votes as soon as he has access to the communication between the voters and the bulletin box (these communications being conducted on the *public* Internet network). Even if we did not find any explicit mention of this, we believe that the designers of the protocol implicitly assume that a corrupted decryption would not be able to control (some of) the communication over the Internet. It should also be noted that a corrupted decryption service could learn the votes even *without access to Internet* if the bulletin box does not shuffle the ballots before sending them. Whether or not shuffling is performed is not completely clear in [15].

*Dishonest bulletin box and receipt generator.* Clearly, if the bulletin box and the receipt generator collude, they can compute $a_1 = a_3 - a_2$ and they can then decrypt all incoming encrypted ballots. More interestingly, a corrupted receipt generator does not need the full cooperation of the bulletin box for breaking ballot secrecy. Indeed, assume that the receipt generator has access, for some voter $V$, to the blinding factor $s_V$ used by the bulletin to blind the ballot. Recall that the receipt generator retrieves $f(o)^{s_V}$ when generating the receipt codes (by computing $\check{w}\check{x}^{-a_3}$). Therefore, the receipt generator can compute $f(o')^{s_V}$ for any possible voting option $o'$. Comparing with the obtained values with $f(o)^{s_V}$ it would easily deduce the chosen option $o$. Of course, the more blinding factors the receipt generator can get, the more voters it can attack. Therefore, the security of the protocol strongly relies on the security of the blinding factors which generation and distribution are left unspecified in the documentation. The bulletin box can also perform a similar attack, provided it can learn some coding function $d_V$ and additionally, provided that it has access to the SMS sent by the receipt generator, which is probably a too strong corruption scenario.

## 6 Further corruption cases using ProVerif

In order to study further corruption cases, we have used ProVerif, the only tool that can analyse equivalence properties in the context of security protocols. Of course, we needed to simplify the equational theory since ProVerif does not handle associative and commutative (AC) symbols and our theory needs four of them. So we have considered the theory $E'$ defined by the equations of Figure 6, except equation (5) that represents homomorphic combination of ciphertexts and we have replaced AC symbols $+$ and $*$ by free function symbols $f$ and $g$. Using this simplified theory, it is clear that we can miss some attacks, but testing corruption assumptions is still relevant even if the attacker is a bit weaker than in our first study.

As ProVerif is designed to prove equivalences between processes that differ only by terms, we need to use another tool, ProSwapper [16], to model the shuffle done by the decryption service. More precisely, we actually used their algorithm to compute directly a shuffle in our ProVerif specification.

The results are displayed in Table 1 and 2 and have been obtained with a standard (old) laptop[1]. In these tables, ✓ indicates that ballot secrecy is satisfied, $\times$ shows that there is an attack, and - indicates that ProVerif was not able to conclude. No indication of times means that we do not proceed to a test in ProVerif but, as we already knew that there was an attack. In particular, all the attacks described in Section 5.2 are displayed in the tables.

[1] 2.00 Ghz processor with 2 GB of RAM Memory

**Table 1.** Results and computation times for the protocol without auditor.

| Corr. Players \ Corr. Voters | 0 | 1 | 2 | 5 | 10 |
|---|---|---|---|---|---|
| None | ✓ 0.4" | ✓ 0.9" | ✓ 2.4" | ✓ 16.1" | ✓ 20'59" |
| Ballot Box (B) | - >1h | | | | |
| Receipt Generator (R) | ✓ 1.1" | ✓ 2.4" | ✓ 5.7" | ✓ 1'15" | ✓ 39'30" |
| Decryption Service (D) | × 0.2" | × | | | |
| B + R | × 0.3" | × | | | |
| D+B, D+R, D+R+B | × | | | | |

**Table 2.** Results and computation times for the protocol with auditor.

| Corr. Players \ Corr. Voters | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| None | ✓ 0.6" | ✓ 1,8" | ✓ 4.1" | ✓ 27.7" | ✓ 11'1" |
| Ballot Box (B) | - >1h | | | | |
| Receipt Generator (R) | ✓ 1.1" | ✓ 1.9" | ✓ 5.9" | ✓ 29.1" | ✓ 10'33" |
| Auditor (A) | ✓ 0.4" | ✓ 1.9" | ✓ 2.6" | ✓ 5.8" | ✓ 12.1" |
| R + A | ✓ 0.6" | ✓ 1.9" | ✓ 5.5" | ✓ 14.5" | ✓ 34.4" |
| B+R, B+R+A, D<br>D + any other combination | × | | | | |

Our case study with ProVerif indicates that ballot secrecy is still preserved even when the Receipt Generator is corrupted (as well as several voters), at least in the simplified theory. Unfortunately, ProVerif was not able to conclude in the case the Ballot Box is corrupted.

## 7 Discussion

We have proposed the first formal proof that the e-voting protocol recently used in Norway indeed satisfies ballot secrecy, even when all but two voters are corrupted and even without the auditor. As expected, ballot secrecy is no longer guaranteed if both the bulletin box and the receipt generator are corrupted. Slightly more surprisingly, the protocol is not secure either if the decryption service is corrupted, as discussed in Section 5.2. More cases of corruption need to be studied, in particular when the bulletin board alone is corrupted, we leave this as future work. In addition, it remains to study other security properties such as coercion-resistance or verifiability. Instead of doing additional (long and technical) proofs, a further step consists in developing a procedure for automatically checking for equivalences. Of course, this is a difficult problem. A first decision procedure has been proposed in [9] but is limited to subterm convergent theories. An implementation has recently been proposed [8] but it does not support such a complex equational theory. An alternative step would be to develop a sound procedure that over-approximate the relation, losing completeness in the spirit of ProVerif [5] but tailored to privacy properties.

We would like to emphasize that the security proofs have been conducted in a symbolic thus abstract model. This provides a first level of certification, ruling out "logical" attacks. However, a full computational proof should be developed. Our symbolic proof can been seen as a first step, identifying the set of messages an attacker can observe when interacting with the protocol. There is however still a long way to go for a computational proof. In particular, it remains to identify which the security assumptions are needed.

It is also important to note that the security of the protocol strongly relies on the way initial secrets are pre-distributed. For example, three private decryption keys $a_1, a_2$, $a_3$ (such that $a_1 + a_2 = a_3$) need to be securely distributed among (respectively) the bulletin board, the receipt generator and the decryptor. Also, a table $(id, s(id))$ containing the blinding factor for each voter needs to be securely distributed to bulletin board and a table $(id, d_{id})$ containing a permutation for each voter needs to be securely distributed to the receipt generator. Moreover, anyone with access with both the codes mailed to the voters and to the SMS emitted by the receipt generator would immediately learn the values of all the votes. We did not find in the documentation how and by who all these secret values were distributed. It should certainly be clarified as it could be a weak point of the system.

## References

1. `http://www.dw-world.de/dw/article/0,,4069101,00.html`.
2. Web page of the norwegian government on the deployment of e-voting. `http://www.regjeringen.no/en/dep/krd/prosjekter/e-vote-2011-project.html`.
3. M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *28th ACM Symposium on Principles of Programming Languages (POPL'01)*, 2001.
4. D. Bernhard, V. Cortier, O. Pereira, B. Smyth, and B. Warinschi. Adapting Helios for provable ballot secrecy. In *16th European Symposium on Research in Computer Security (ESORICS'11)*, LNCS. Springer, 2011.
5. B. Blanchet. An automatic security protocol verifier based on resolution theorem proving (invited tutorial). In *20th International Conference on Automated Deduction (CADE-20)*, Tallinn, Estonia, July 2005.
6. B. Blanchet, M. Abadi, and C. Fournet. Automated verification of selected equivalences for security protocols. In *20th IEEE Symposium on Logic in Computer Science (LICS 2005)*, pages 331–340. IEEE Computer Society, June 2005.

7. R. Chadha, Ş. Ciobâcă, and S. Kremer. Automated verification of equivalence properties of cryptographic protocols. In *21th European Symposium on Programming (ESOP'12)*, LNCS. Springer, 2012. To appear.
8. V. Cheval, H. Comon-Lundh, and S. Delaune. Trace equivalence decision: Negative tests and non-determinism. In *18th ACM Conference on Computer and Communications Security (CCS'11)*. ACM Press, Oct. 2011. To appear.
9. V. Cortier and S. Delaune. A method for proving observational equivalence. In *22nd Computer Security Foundations Symposium (CSF'09)*. IEEE Computer Society, 2009.
10. V. Cortier and B. Smyth. Attacking and fixing Helios: An analysis of ballot secrecy. In *24th Computer Security Foundations Symposium (CSF'11)*. IEEE Computer Society, 2011.
11. V. Cortier and C. Wiedling. A formal analysis of the Norwegian e-voting protocol. Technical Report RR-7781, INRIA, Nov. 2011.
12. S. Delaune, S. Kremer, and M. D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, 2009.
13. A. J. Feldman, J. A. Halderman, and E. W. Felten. Security analysis of the diebold accuvote-ts voting machine. `http://itpolicy.princeton.edu/voting/`, 2006.
14. A. Fujioka, T. Okamoto, and K. Ohta. A Practical Secret Voting Scheme for Large Scale Elections. In *AUSCRYPT'92: Workshop on the Theory and Application of Cryptographic Techniques*, LNCS. Springer, 1992.
15. K. Gjøsteen. Analysis of an internet voting protocol. Cryptology ePrint Archive, Report 2010/380, 2010. `http://eprint.iacr.org/`.
16. P. Klus, B. Smyth, and M. D. Ryan. ProSwapper: Improved equivalence verifier for ProVerif. `http://www.bensmyth.com/proswapper.php`, 2010.
17. S. Kremer, M. D. Ryan, and B. Smyth. Election verifiability in electronic voting protocols. In *15th European Symposium on Research in Computer Security (ESORICS'10)*, LNCS. Springer, 2010.
18. R. Küsters and T. Truderung. An Epistemic Approach to Coercion-Resistance for Electronic Voting Protocols. In *IEEE Symposium on Security and Privacy (S&P 2009)*, pages 251–266. IEEE Computer Society, 2009.
19. R. Küsters, T. Truderung, and A. Vogt. A Game-Based Definition of Coercion-Resistance and its Applications. In *23nd IEEE Computer Security Foundations Symposium (CSF 2010)*. IEEE Computer Society, 2010.
20. R. Küsters, T. Truderung, and A. Vogt. Proving Coercion-Resistance of Scantegrity II. In *12th International Conference on Information and Communications Security (ICICS 2010)*, volume 6476 of *LNCS*. Springer, 2010.
21. B. Lee, C. Boyd, E. Dawson, K. Kim, J. Yang, and S. Yoo. Providing Receipt-Freeness in Mixnet-Based Voting Protocols. In *6th International Conference on Information Security and Cryptology (ICISC'03)*, volume 2971 of *LNCS*, pages 245–258. Springer, 2004.
22. J. Liu. A Proof of Coincidence of Labeled Bisimilarity and Observational Equivalence in Applied Pi Calculus. `http://lcs.ios.ac.cn/~jliu/papers/LiuJia0608.pdf`, 2011.
23. T. Okamoto. Receipt-Free Electronic Voting Schemes for Large Scale Elections. In *5th International Workshop on Security Protocols (SP'97)*, volume 1361 of *LNCS*, pages 25–35. Springer, 1998.
24. S. Wolchok, E. Wustrow, J. A. Halderman, H. K. Prasad, A. Kankipati, S. K. Sakhamuri, V. Yagati, and R. Gonggrijp. Security analysis of india's electronic voting machines. In *17th ACM Conference on Computer and Communications Security (CCS 2010)*, 2010.