Journal of Computer Security 25 (2017) 21–57
DOI 10.3233/JCS-15777
IOS Press

A formal analysis of the Norwegian E-voting protocol

Véronique Cortier [a] and Cyrille Wiedling [b,*]
[a] *CNRS-LORIA, Nancy, France*
[b] *INRIA, Rennes, France*

Abstract. Norway used e-voting in its last political election both in September 2011 and September 2013, with more than 28,000 voters using the e-voting option in 2011, and 70,000 in 2013. While some other countries use a black-box, proprietary voting solution, Norway has made its system publicly available. The underlying protocol, designed by Scytl, involves several authorities (a ballot box, 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. In this paper, we propose a formal analysis of the protocol used in Norway, w.r.t. ballot secrecy, considering several corruption scenarios. We use a state-of-the-art definition of ballot secrecy, based on equivalence properties and stated in the applied pi-calculus.

Keywords: Ballot secrecy, formal methods, e-voting

Used in September 2011 and September 2013 for municipality and county elections in Norway [25], e-voting was tested in ten municipalities. During this nationwide local elections, more than 28,000 voters did use internet to cast their vote in 2011 and 70,000 in 2013. While many countries use black-box proprietary solutions, Norway made the protocol publicly available [23]. One key feature of this system is that voters can check that their votes have correctly reached the ballot box ("cast-as-intended" property) without anyone else knowing their vote. The goal of this paper is to conduct a thorough analysis of the Norwegian protocol for the ballot secrecy property: does this system guarantee that no one can know how some voter voted?

Formal methods have been successfully applied to security protocols with the development of several tools such as ProVerif [10], Avispa [4], or Scyther [20] that can automatically analyse both protocols of the literature and fully deployed protocols such as TLS [9]. We therefore chose to model and analyse the Norwegian protocols in a symbolic model named the applied pi-calculus model [1]. A first issue comes from the fact that the underlying encryption primitive is non standard and rather complex. Indeed, the decryption key is split in two shares a_1 and a_2 with $\mathsf{a}_3 = \mathsf{a}_1 + \mathsf{a}_2$. Each of these three keys $\mathsf{a}_1, \mathsf{a}_2, \mathsf{a}_3$ is given to one administration authority. As further explained in Section 1, this allows for re-encryption and is crucial for the "cast-as-intended" property of the protocol.

Contributions. Our first contribution is a detailed symbolic model for this particular encryption primitive. More precisely, we provide a rewriting system that models El Gamal encryption, re-encryption, blinding function, signatures and zero-knowledge proofs reflecting the primitives used in the protocol. Then, we model the whole system as a process in the applied pi-calculus model. Our second main contribution is a proof of ballot secrecy for several corruption scenarios. Ballot secrecy is typically modeled as follows [21]: an attacker should not be able to distinguish the case where Alice is voting *a* and Bob

[*] Corresponding author. E-mail: cyrille.wiedling@inria.fr.

is voting b from the converse scenario where Alice is voting b while Bob is voting a. Such a property is typically described by an equivalence of the form

$$\mathsf{Alice}(a) \mid \mathsf{Bob}(b) \approx \mathsf{Alice}(b) \mid \mathsf{Bob}(a)$$

where $\mathsf{Alice}(v)$ represents voter Alice voting for v. Such indistinguishability properties are formalized through behavioral equivalence. Here we use observational equivalence $\approx$ as defined in [1]. Combined with complex equational theories (in particular equational theories with associative and commutative – AC – operators), no existing tool can check for equivalence. Indeed, dedicated tools such as SPEC [34], APTE [15], or AkisS [13] can only handle standard primitives, with the exception of AkisS which covers more primitives but no AC operators. All three tools are moreover limited to a fixed (and small) number of sessions. Recently, the tool Tamarin has been enhanced to cope with equivalence properties [5] but for the moment it requires a high level of interactions. Therefore, the only natural candidate for an automatic analysis of ballot secrecy is the tool ProVerif [10], one of the most generic tools for security protocols. It can check for equivalence [12] for an unbounded number of sessions. However, ProVerif cannot handle equational theories with AC operators either. [2] devises a technique that transforms an equational theory with AC operators into an equational theory without AC operators, that ProVerif can handle. However, [2] is particular to the re-encryption theory (together with any non AC theory) and it is unclear whether it can be adapted to the complex equational theory needed for the Norwegian protocol. Thus we conducted a preliminary analysis of the Norwegian protocol for a simplified – more abstract – model of the protocol (without AC operators), and finite number of voters.

In order to obtain stronger security guarantees, the full equational theory of the cryptographic primitives should be considered. Therefore, the main contribution of the paper is a proof "by hand" of ballot secrecy for two main corruption scenarios: when all authorities are honest (but all but two voters are corrupted) and when the ballot box is corrupted (and again all but two voters are corrupted). We believe that ballot secrecy can be established in a similar way when the receipt generator is corrupted. A preliminary version of ballot secrecy under the first corruption scenario was presented in [18]. While the encryption scheme used in the Norwegian system is particular to the protocol, the Norwegian system also make use of more standard primitives such as signatures or zero-knowledge proofs. When proving ballot secrecy, we developed generic lemmas that could be re-used in subsequent works (e.g. [3]).

Related Work. Since our initial study of the Norwegian protocol [18], Gjøsteen has proposed a detailed security analysis of the protocol [24] for both ballot secrecy and verifiability, under several trust assumptions (a first model and security definitions already appeared in [23]). A first main difference between the two approaches is the security model: we consider a symbolic model while [23,24] rely on a, more concrete, computational model where the attacker is any polynomial probabilistic Turing machine. In that respect, the study in [23,24] is less abstract and provides more precise security assumptions on the underlying primitives. In contrast, one advantage of symbolic models is that they are more amendable for automation as demonstrated by our use of ProVerif for a simplified model. Ballot secrecy is formalized in two different ways in these two approaches: here, we use a definition of ballot secrecy that is the standard ballot secrecy definition in symbolic models [21]. [23,24] model ballot secrecy (and verifiability) through an ideal functionality that needs to be adapted to the protocol under consideration. Another difference between [23,24] and this work lies in the trust assumptions and the properties that are considered. [23,24] consider both ballot secrecy and verifiability while we focus here on ballot secrecy. Regarding trust assumptions and as it is often the case in other studies of e-voting protocols, at least in symbolic models, we assume that the communications between the voter's computer and the server can be eavesdropped by an attacker, although in practice, these communications typically happen under some secure channel (e.g. TLS). Whenever possible, it is interesting to prove security without relying on external secure channels since their security is typically not under the control of the authorities, as recently exemplified in an Australian election [26]. In contrast, [23,24] assume private communications between the voter and the server and consider all the cases where one of the authorities is corrupted (while we do not consider the case of a corrupted receipt generator). Interestingly, [23,24] shows that the Norwegian protocol remains secure when the decryption device is corrupted while we show this is not the case. This indicates that, when the decryption device is corrupted, ballot secrecy solely relies on the security of the secure channels used by voters.

Several other e-voting protocols have been studied using formal methods. The FOO [22], Okamoto [32], and Lee *et al.* [30] voting protocols have been analysed in [21]. Similarly, Helios has been recently proved secure both in a formal [17] and a computational [7,8] model. Helios is actually an implementation of a voting system proposed and analyzed (for the available definitions at that time) by Cramer *et al* [19]. All these protocols were significantly simpler to analyse in a symbolic model due to the fact that the cryptographic primitives were easier to abstract as a term algebra and due to the fact that these protocols involve less steps. Civitas has been analyzed in [29] in a symbolic model, for a rather rich equational theory. The analysis of this protocol remains simpler than the case of the Norwegian protocol, due to the fact that in Civitas, the voting phase does not involve any interaction with the bulletin board. Our study (together with [23,24]) forms the first security proof of a fully deployed Internet protocol in politically binding elections. Enlarging the scope to voting systems that may take place in polling stations (that is not just Internet voting), security analyses include Scantegrity II [14,29,33], Prêt-à-voter [27], and STAR-Vote [6].

Outline of the paper. We provide an informal description of the protocol in Section 1, including details about the different phases of the voting process. Section 3 presents our formal model of the protocol in the applied pi-calculus. The protocol makes use of a special encryption function in combination with signatures, zero-knowledge proofs, blinding functions, and coding functions. We therefore propose a new equational theory reflecting the unusual behavior of the primitives. The main results and corresponding proofs are presented in Section 4. Section 5 gathers the main lemmas needed for the proofs. We believe these lemmas to be of independent interest in the sense they can be useful for further formal studies of different protocols. Most of the proofs are detailed in Sections F and 6 with some additional lemmas postponed to the appendix. Finally, in Section 7, we present a simplified model of the Norwegian protocol and the corresponding security analysis using ProVerif.

1. The Norwegian e-voting protocol

The Norwegian protocol features four players that define 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 to be authenticated channels, channels between infrastructure's player are untappable, and channels between voters and receipt generator are unidirectional out-of-band channels. (Example of SMS is given in [23].) 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 the auditor verifies the correctness of the election.

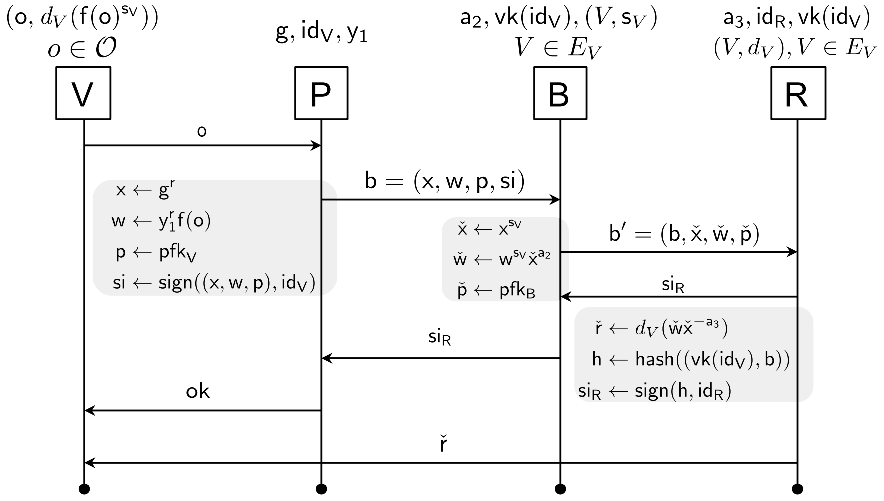
$(\mathsf{o}, d_V(\mathsf{f}(\mathsf{o})^{\mathsf{s_V}}))$
$o \in \mathcal{O}$
$\mathsf{g}, \mathsf{id_V}, \mathsf{y_1}$
$\mathsf{a_2}, \mathsf{vk}(\mathsf{id_V}), (V, \mathsf{s}_V)$
$V \in E_V$
$\mathsf{a_3}, \mathsf{id_R}, \mathsf{vk}(\mathsf{id_V})$
$(V, d_V), V \in E_V$
V
P
B
R
o
$\mathsf{x} \leftarrow \mathsf{g^r}$
$\mathsf{w} \leftarrow \mathsf{y_1^r f(o)}$
$\mathsf{p} \leftarrow \mathsf{pfk_V}$
$\mathsf{si} \leftarrow \mathsf{sign}((\mathsf{x}, \mathsf{w}, \mathsf{p}), \mathsf{id_V})$
$\mathsf{b} = (\mathsf{x}, \mathsf{w}, \mathsf{p}, \mathsf{si})$
$\check{\mathsf{x}} \leftarrow \mathsf{x^{s_V}}$
$\check{\mathsf{w}} \leftarrow \mathsf{w^{s_V}} \check{\mathsf{x}}^{\mathsf{a_2}}$
$\check{\mathsf{p}} \leftarrow \mathsf{pfk_B}$
$\mathsf{b'} = (\mathsf{b}, \check{\mathsf{x}}, \check{\mathsf{w}}, \check{\mathsf{p}})$
$\mathsf{si_R}$
$\check{\mathsf{r}} \leftarrow d_V(\check{\mathsf{w}}\check{\mathsf{x}}^{-\mathsf{a_3}})$
$\mathsf{h} \leftarrow \mathsf{hash}((\mathsf{vk}(\mathsf{id_V}), \mathsf{b}))$
$\mathsf{si_R} \leftarrow \mathsf{sign}(\mathsf{h}, \mathsf{id_R})$
$\mathsf{si_R}$
ok
$\check{\mathsf{r}}$

Fig. 1. Submission of one vote.

1.1. Setting phase

Before the election, a finite cyclic group G of some prime order q and generator by g is selected, three private keys $\mathsf{a_1}$, $\mathsf{a_2}$, and $\mathsf{a_3}$ (such that $\mathsf{a_1} + \mathsf{a_2} \equiv \mathsf{a_3}[q]$) are generated and distributed over respectively D, B, and R, while the corresponding public keys $\mathsf{y_1} = \mathsf{g^{a_1}}$, $\mathsf{y_2} = \mathsf{g^{a_2}}$ and $\mathsf{y_3} = \mathsf{g^{a_3}}$ are made publicly available Each voter V is assumed to have a signing key $\mathsf{id_V}$, with a corresponding verification key, $\mathsf{vk}(\mathsf{id_V})$, which is public. The Ballot box B is provided with a table $V \mapsto \mathsf{s_V}$ that associates each voter with a blinding factor $\mathsf{s_V}$. The Receipt generator R is also assumed to have a signing key $\mathsf{id_R}$, with a corresponding public verification key, $\mathsf{vk}(\mathsf{id_R})$; and it is given, for each voter V, a pseudo-random function $d_V : G \to \mathcal{C}$, where $\mathcal{C}$ is the set of receipts. Finally, each voter V is assumed to receive by surface mail a table that associates to any voting option o (from the set $\mathcal{O}$), a precomputed receipt code $d_V(\mathsf{f}(\mathsf{o})^{\mathsf{s_V}}) \in \mathcal{C}$, where $\mathsf{f} : \mathcal{O} \to G$ is an injective encoding function.

1.2. Submission phase

The submission phase is summarized in Fig. 1. We detail in this section 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 $\check{\mathsf{r}}$ sent by the receipt generator through some out-of-band channel (typically a SMS message). Using the receipt, he verifies that the correct vote was submitted, that is, he checks that $\check{\mathsf{r}} = d_V(\mathsf{f}(\mathsf{o})^{\mathsf{s_V}})$ by verifying that the receipt code $\check{\mathsf{r}}$ indeed appears in the line associated to the voting option o he has chosen.

This check ensures in particular the "cast-as-intended" property. In case the voter's computer is corrupted and encrypts another vote (say the computer wishes to cast a vote for the Pirate party) then this would be eventually discovered by the voter when receiving the receipt.

Computer (P). Voter's computer encrypts voter's ballot with the public key $\mathsf{y_1}$ using standard El Gamal encryption. The resulting ballot is $\langle \mathsf{g^r}, \mathsf{y_1^r f(o)}\rangle$. P also proves that the resulting ciphertext corresponds to a valid voting option, by computing a standard proof of knowledge $\mathsf{pfk_V}$. (This proof is formally presented in Section 3, but a detailed description can be found in [23].) P also signs, on the behalf of the Voter, 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.

Ballot box (B). Upon receiving an encrypted and signed ballot b from a computer, the Ballot box first checks the correctness of signatures and proofs before re-encrypting the original encrypted ballot with $\mathsf{a_2}$ and blinding it with $\mathsf{s_V}$. B also generates a proof $\mathsf{pfk_B}$, 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 R, it simply checks that the Receipt generator's signature is valid, and sends it to the computer.

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

- a receipt code $\check{\mathsf{r}} = d_V(\check{\mathsf{w}}\check{\mathsf{x}}^{-\mathsf{a_3}})$ sent by out-of-band channel directly to the Voter. Intuitively, the Receipt generator decrypts the (blinded) ballot, applying the function d_V associated to the voter. This receipt code gives assurance to the voter that the correct vote was submitted to the Ballot box.
- a signature on a hash of the original encrypted ballot for the Ballot box. Once transmitted by B, it is checked and passed on to the Voter's Computer, which checks it once more and informs the Voter that his vote has been taken into account.

1.3. Counting phase

Once the voting phase is over, the counting phase begins (Fig. 2). The Ballot box selects the encrypted votes $x_1, \ldots, x_k$ which need to be decrypted (if a voter has voted several times, all the submitted ballots remain in the memory of the Ballot box but only the last ballot should be sent) and sends them to the Decryption service. The whole content of the Ballot box $\mathsf{b_1}, \ldots, \mathsf{b_n}$ ($n \geqslant k$) 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 shuffles the decrypted votes before publishing them. It therefore outputs a message of the form $\mathsf{dec}(x_{\sigma(1)}, \mathsf{a_1}), \ldots, \mathsf{dec}(x_{\sigma(k)}, \mathsf{a_1})$ where σ denotes the permutation obtained by shuffling the votes. It also provides the Auditor with a proof $\mathsf{pfk_D}$ 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 it computes its own list of encrypted ballots which should be counted. He compares this list with the one received from the Decryption service and checks the proof provided by the latter.

1.4. Security analysis

The rest of the paper is devoted to the modeling and analysis of the protocol. We summarize here informally our main results and we list our main assumptions and simplifications. We formally prove ballot secrecy under two threat scenario:

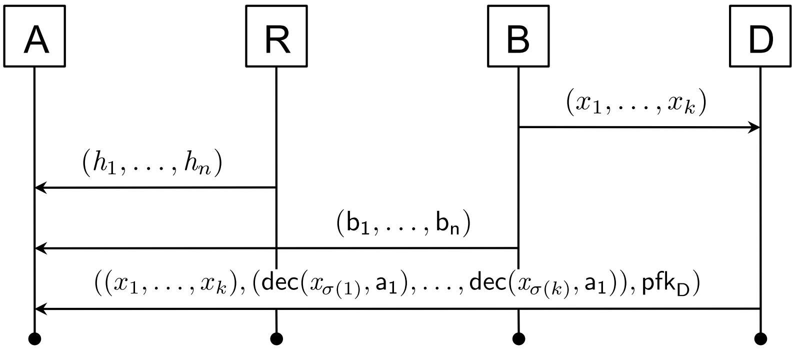
A
R
B
D
$(x_1, \ldots, x_k)$
$(h_1, \ldots, h_n)$
$(\mathsf{b}_1, \ldots, \mathsf{b}_n)$
$((x_1, \ldots, x_k), (\mathsf{dec}(x_{\sigma(1)}, \mathsf{a}_1), \ldots, \mathsf{dec}(x_{\sigma(k)}, \mathsf{a}_1)), \mathsf{pfk}_\mathsf{D})$

Fig. 2. Counting phase.

– All but two voters are dishonest and all the authorities (the Ballot box, the Receipt generator, the Decryption service and the Auditor) are honest.
– All but two voters are dishonest and all the authorities but the Ballot box are honest.

These two results are obtained assuming authenticated but not necessarily secure communication channels between voters and the Ballot box and Receipt generator, meaning that an attacker may eavesdrop the communications. For this reason, ballot secrecy does not hold as soon as the Decryption device is corrupted. Note also that ballot privacy is also broken as soon as the voter's computer (P) is corrupted since the voter sends her vote to her computer. We therefore had to assume P to be honest and we chose to model it together with the voter's behavior. Authentication between the voter and the Ballot box is ensured through a login and password mechanism while authentication from the receipt generator to the voter relies on the out-of-band channel used between them (typically a SMS message).

When modeling the Norwegian protocol in a symbolic model, we had to proceed to some simplifications. First, and as discussed later in Section 3.1, the cryptographic primitives are abstracted by terms together with an equational theory, which potentially leaves out some flaws due to some crafty modifications of some messages. Second, we chose not to model revoting, for simplicity but also since it is explicitly and strongly discouraged in [23], as it may provide to an attacker that control the Ballot box the opportunity to swap the initial and the resubmitted votes. Third, for simplicity, we only consider one-option votes (that voters select one option), rather that a vector of options as in [23]. Finally, we also omit the proof of correct decryption provided by the decryption device since it should not affect ballot secrecy.

2. Applied pi-calculus

We describe here the applied pi-calculus [1], introduced by M. Abadi and C. Fournet. The applied pi-calculus is a process algebra that is often used to model protocols, and we briefly recall here the notations and definitions, providing some examples.

2.1. Terms

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* Σ consisting of a finite set of *function symbols* (to represent cryptographic primitives). A function symbol f is assumed to be given with its arity $\mathsf{ar}(f)$. Then, the set of terms $T(\Sigma, \mathcal{X}, \mathcal{N})$ is formally defined by the following grammar:

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

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 σ to the free variables of the term N. A term is called *ground* when it does not contain variables.

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

Example 1. A signature for symmetric, asymmetric encryption and signature is

$$\Sigma = \{\mathsf{checksign}, \mathsf{dec}, \mathsf{pdec}, \mathsf{enc}, \mathsf{penc}, \mathsf{sign}, \mathsf{pk}\}$$

where penc and pdec represent resp. asymmetric (randomized) encryption and decryption, enc and dec stand resp. for symmetric (randomized) encryption and decryption, sign models signature, checksign represents a function that checks the validity of signature, and pk represents the public key associated to a secret key. For example, the term $\mathsf{penc}(m, r, \mathsf{pk}(sk_a))$ represents the asymmetric encryption of the message m with the public key corresponding to the secret key sk_a and random factor r. The properties of the cryptographic functions are represented by the equational theory E_{enc}:

$$\mathsf{dec}\big(\mathsf{enc}(m, r, k), k\big) = m$$

$$\mathsf{pdec}\big(\mathsf{penc}\big(m, r, \mathsf{pk}(sk)\big), sk\big) = m$$

$$\mathsf{checksign}\big(\mathsf{sign}(m, sk), \mathsf{pk}(sk), m\big) = \mathsf{ok}.$$

The first equation models that symmetric decryption is only successful if the key used for decryption is the same as the one used for encryption. The second equation reflects that asymmetric decryption only succeeds when the corresponding secret key is used. Finally, the last equation checks that a signature corresponds to a given message and a given verification key.

For some cryptographic primitives, such as homomorphic encryption, it is necessary to introduce associative and commutative symbols. Equational theories including such symbols are called *AC-theories*. Therefore we define an equality modulo AC, noted $=_{AC}$, which denotes that two terms are syntactically equal modulo the associative or commutative properties for each AC symbol $+$.

$$x + (y + z) = (x + y) + z$$

$$x + y = y + x$$

Example 2. To represent homomorphic encryption, we may use the signature $\Sigma = \{+, *, \mathsf{enc}\}$ with $+$ and $*$ two associative and commutative (AC) symbols and the corresponding equational theory E_{AC}, which includes the associative and commutative properties of $+$ and $*$ symbols, and the equation:

$$\mathsf{enc}(m, k) * \mathsf{enc}(n, k) = \mathsf{enc}(m + n, k).$$

This equation models that two ciphertexts encrypted with the same key can be combined to create a ciphertext which corresponding plaintext is the sum of the two previous plaintexts.

2.2. *Rewriting system*

It might be difficult to work modulo an equational theory. Instead, it is often possible (and more convenient) to reason with a rewriting system. Formally, a *rewriting system* $\mathcal{R}$ is a set of *rewriting rules* of the form $l \rightarrow r$ with l and r two terms. We say that a term s is *rewritten* in t for rule $l \rightarrow r$, noted $s \rightarrow t$, if there exists a position p in s and a substitution θ such that $s|_p = l\theta$ and $t = s[r\theta]_p$.

Definition 1 (Convergence)**.** A rewriting system $\mathcal{R}$ is said *convergent* if:

- for every ground term U, there exists no infinite sequence $U \rightarrow U_1 \rightarrow \cdots \rightarrow U_k \rightarrow \cdots$. (In this case, we say that $\mathcal{R}$ is terminating.)
- for every ground terms U, U_1, and U_2 such that $U \rightarrow^* U_1$ and $U \rightarrow^* U_2$, there exists V such that $U_1 \rightarrow^* V$, and $U_2 \rightarrow V$. (In this case, we say that $\mathcal{R}$ is confluent.)

For AC-theories, we consider rewriting systems modulo AC. Formally, a term s is rewritten modulo AC in t, noted $s \rightarrow_{AC} t$, for a rule $l \rightarrow r$ if there exist s' and t' two terms such that $s =_{AC} s'$, $t =_{AC} t'$ and s' is rewritten in t' for the rule $l \rightarrow r$. Then, it is possible to define the notion of AC-convergence.

Definition 2 (AC-Convergence)**.** A rewriting system $\mathcal{R}$ is said AC-convergent if:

- for every ground term U, there is no infinite sequence $U \rightarrow_{AC} U_1 \rightarrow_{AC} \cdots \rightarrow_{AC} U_k \cdots$. (In this case, we say that $\mathcal{R}$ is AC-terminating.)
- for every ground terms U, U_1, and U_2 such that $U \rightarrow^*_{AC} U_1$ and $U \rightarrow^*_{AC} U_2$, there exists V such that $U_1 \rightarrow^*_{AC} V$, and $U_2 \rightarrow^*_{AC} V$. (In this case, we say that $\mathcal{R}$ is AC-confluent.)

2.3. *Processes*

Processes and *extended processes* are defined in Fig. 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 parallel composition of P with itself). $\nu n.P$ creates a fresh name n and then behaves like P. The process if ϕ then P else Q behaves like P if ϕ 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 $\nu u_1.\ldots.\nu u_n$. 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 $\nu x(\{^M/_x\} \mid P)$ corresponds exactly to let $x = M$ in P.

$P, Q, R ::=$	(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
$A, B, C ::=$	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. 3. Syntax for processes.

As in [17], we slightly extend the applied pi-calculus by letting conditional branches now depend on formulae defined by the following grammar:

$$\phi, \psi ::= M = N \mid M \neq N \mid \phi \wedge \psi$$

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 is delimited by binders $u(x)$ and νu. 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 $\mathsf{fn}(M)$ (resp. $\mathsf{fv}(M)$) for the set of names (resp. 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 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.

We refer the reader to Section 2.6 for a full example.

2.4. 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 Fig. 4. It is closed by α-conversion of both bound names and bound variables, and closed under application of evaluation contexts. Structural equivalence corresponds

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 $u \notin \mathsf{fv}(A) \cup \mathsf{fn}(A)$
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.

to some structural rewriting that does not change the semantics of a process. The *internal reductions* and *labelled reductions* are defined in Fig. 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.

2.5. *Equivalences*

Privacy properties are often stated as equivalence relations [21]. 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 the indistinguishability of two sequences of terms.

Definition 3 (Static equivalence)**.** Two closed frames φ and ψ are statically equivalent, denoted $\varphi \approx_s \psi$, if $\mathsf{dom}(\varphi) = \mathsf{dom}(\psi)$ and there exists a set of names $\tilde{n}$ and substitutions σ, τ such that $\varphi \equiv \nu\tilde{n}.\sigma$ and $\psi \equiv \nu\tilde{n}.\tau$ and for all terms M, N such that $\tilde{n} \cap (\mathsf{fn}(M) \cup \mathsf{fn}(N)) = \emptyset$, 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, $\varphi(A) \approx_s \varphi(B)$.

Intuitively, two sequences of messages φ and ψ are distinguishable to an attacker (i.e. they are not statically equivalent) if the attacker can build a public test $M = N$ that holds for φ but not for ψ (or the converse).

Example 3. Consider the signature and equational theory E_{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 have $\mathsf{penc}(s_1, r_1, x_2)\sigma_1 =_E x_1\sigma_1$ but $\mathsf{penc}(s_1, r_1, x_2)\sigma_2 \neq_E x_1\sigma_2$.

Intuitively, since the randomness of the encryption is public, an attacker may reconstruct the ciphertexts and compare. The two messages become indistinguishable as soon as the randomness remain private. That is, we have that $\nu(k, r_1).\sigma_1 \approx_s \nu(k, r_2).\sigma_2$.

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

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

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

where α 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.

all contexts (i.e. all adversaries), which renders the proofs difficult. Since observational equivalence has been shown to coincide [1,31] with labelled bisimilarity, we adopt the latter in the remaining of the paper.

Definition 4 (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 $\mathsf{fv}(\alpha) \subseteq \mathsf{dom}(A)$ and $\mathsf{bn}(\alpha) \cap \mathsf{fn}(B) = \emptyset$, then $B \to^* \xrightarrow{\alpha} \to^* B'$ and $A'\mathcal{R}B'$ for some B'.

Intuitively, two processes A and B are labelled bisimilar if, anyhow the process A (resp. B) behaves, the process B (resp. A) may behave with the same visible actions and such that their resulting frames are statically equivalent, that is, an attacker cannot distinguish between them.

2.6. A detailed example

We provide a detailed example to illustrate the syntax and semantics of the applied pi-calculus. Readers already familiar with this formalism may skip this section. Let us consider a simple protocol proposed (for illustration purposes) by B. Blanchet [11].

$$\begin{tikzcd}
\text{Alice} \arrow[r, shift left, "\{\mathsf{sign}(k,\mathsf{sk_A})\}_{\mathsf{pk_B}}"] & \text{Bob} \arrow[l, shift left, "\{s\}_k"]
\end{tikzcd}$$

In this protocol, Alice sends a newly generated and signed key k to Bob, the whole message encrypted using his public key $\mathsf{pk_B}$. Then, the receiver (Bob) checks that the signature belongs to the intended sender (Alice). He then encrypts a fresh secret s with k using symmetric encryption and sends it to Alice.

To model this protocol in applied-pi calculus, we use the equational theory E_{enc} described in Example 1 that models the properties of the different primitives used in the protocol. The behavior of the sender and receiver are modeled in the applied pi-calculus as follows.

$$\begin{aligned}
&A := \nu k.\nu r_a \,.\, \overline{c}\langle \mathsf{penc}(\mathsf{sign}(k, sk_a), r_a, \mathsf{pk_B})\rangle \,.\, c(x_1)\\
&B(s) := c(x_2) \,.\, B'(s)\\
&B'(s) := \text{let } y = \mathsf{pdec}(x_2, sk_b) \text{ in}\\
&\quad \text{if } \mathsf{checksign}(y, \mathsf{pk_A}) = \mathsf{ok} \text{ then } B''(s)\\
&B''(s) := \nu r_b \,.\, \overline{c}\langle \mathsf{enc}(s, r_b, k)\rangle
\end{aligned}$$

In these processes, a and b are secret values that correspond to the private key pairs of Alice and Bob, respectively. Since Alice should not have a direct access to b, the secret key of Bob, A is given access to Bob's public key through a variable $\mathsf{pk_B}$ and similarly for B. The whole protocol can be easily expressed using Alice and Bob processes:

$$P(s) := \nu(a, b) \,.\big[A \mid B(s) \mid \Gamma\big]$$

where $\Gamma = \{^{\mathsf{pk}(sk_a)}/_{\mathsf{pk_B}}, {}^{\mathsf{pk}(sk_b)}/_{\mathsf{pk_B}}\}$. As mentioned above, sk_a and sk_b are protected but since $\mathsf{pk}(sk_a)$ and $\mathsf{pk}(sk_b)$ are public keys, we published them in a frame implying that they are available to anyone (including the attacker). We first illustrate the semantics of the internal reductions by simulating the normal execution of the protocol, without any interference.

$$\begin{aligned}
P(s) &\xrightarrow{\text{(COMM)}} \nu(sk_a, sk_b, k, r_a).\big[c(x_1) \mid B'(s) \mid \{^{\mathsf{penc}(\mathsf{sign}(k,sk_a),r_a,\mathsf{pk_B})}/_{x_2}\} \mid \Gamma\big]\\
&\xrightarrow{\text{(THEN)}} \nu(sk_a, sk_b, k, r_a).\big[c(x_1) \mid B''(s) \mid \{^{\mathsf{penc}(\mathsf{sign}(k,sk_a),r_a,\mathsf{pk_B})}/_{x_2}\} \mid \Gamma\big]\\
&\xrightarrow{\text{(COMM)}} \nu\tilde{n}.\big[\{^{\mathsf{enc}(s,r_b,k)}/_{x_1}, {}^{\mathsf{penc}(\mathsf{sign}(k,sk_a),r_a,\mathsf{pk_B})}/_{x_2}\} \mid \Gamma\big], \quad \text{with } \tilde{n} = (sk_a, sk_b, k, r_a, r_b).
\end{aligned}$$

Alice sends her signed and encrypted message, which is received by Bob.

Actually, this simple protocol is flawed. Indeed, an intruder may impersonate Alice's identity from Bob's point of view. This attack only requires that Alice had, once in the past, spoken to the intruder using this protocol. This can be showed using our applied-pi calculus model by adding

$\mathsf{penc}(\mathsf{sign}(k_c, sk_a), r_c, \mathsf{pk_C})$ to the initial frame, which is a message that Alice would have sent to Charlie according to the protocol. Let us define this slightly new process:

$$P_c(s) := \nu(sk_a, sk_b, sk_c, r_c, k_c)\ .\big[A \mid B(s) \mid \Gamma_c\big],$$

where $\Gamma_c = \{^{\mathsf{penc}(\mathsf{sign}(k_c,sk_a),r_c,\mathsf{pk_C})}/_{x}, {}^{\mathsf{pk}(a)}/_{\mathsf{pk_A}}, {}^{\mathsf{pk}(b)}/_{\mathsf{pk_B}}, {}^{\mathsf{pk}(sk_c)}/_{\mathsf{pk_C}}\}$. Let us consider now the following execution: Charlie can send a message $M = \mathsf{penc}(\mathsf{pdec}(x, \mathsf{sk_C}), r', \mathsf{pk_B})$ to Bob, where x is the message previously received from Alice.

$$\begin{aligned}
P_c(s) &\xrightarrow{c(M)} \nu(sk_a, sk_b, r_c, k_c).\big[A \mid B' \mid \Gamma_c\big]\\
&\xrightarrow{\text{(THEN)}} \nu(sk_a, sk_b, r_c, k_c).\big[A \mid B'' \mid \Gamma_c\big]\\
&\xrightarrow{\nu x_c.\bar{c}\langle x_c\rangle} \nu(sk_a, sk_b, r_c, k_c, r_b).\big[A \mid \{^{\mathsf{enc}(s,r_b,k_c)}/_{x_c}\} \mid \Gamma_c\big].
\end{aligned}$$

Let us define the substitution $\sigma = \{^{\mathsf{enc}(s,r_b,k_c)}/_{x_c}, {}^{M}/_{x_2}\} \mid \Gamma_c$. In this execution, the test performed by B succeeds since, according to the equation theory, we have:

$$\begin{aligned}
M\sigma &= \mathsf{penc}\big(\mathsf{pdec}(x, \mathsf{sk}(c)), r', \mathsf{pk_B}\big)\sigma\\
&= \mathsf{penc}\big(\mathsf{pdec}\big(\mathsf{penc}\big(\mathsf{sign}(k_c, \mathsf{sk}(a)), r_c, \mathsf{pk}(c)\big), \mathsf{sk}(c)\big), r', \mathsf{pk}(b)\big)\\
&=_{E_{\mathsf{enc}}} \mathsf{penc}\big(\mathsf{sign}(k_c, \mathsf{sk}(a)), r', \mathsf{pk}(b)\big),
\end{aligned}$$

thus,

$$\begin{aligned}
&\mathsf{checksign}\big(\mathsf{pdec}\big(x_2, \mathsf{sk}(b)\big), \mathsf{pk_A}\big)\sigma\\
&\quad =_{E_{\mathsf{enc}}} \mathsf{checksign}\big(\mathsf{pdec}\big(\mathsf{penc}\big(\mathsf{sign}(k_c, \mathsf{sk}(a)), r', \mathsf{pk}(b)\big), \mathsf{sk}(b)\big), \mathsf{pk}(a)\big)\\
&\quad =_{E_{\mathsf{enc}}} \mathsf{checksign}\big(\mathsf{sign}(k_c, \mathsf{sk}(a)), \mathsf{pk}(a)\big)\\
&\quad =_{E_{\mathsf{enc}}} \mathsf{ok}
\end{aligned}$$

which implies that the message is accepted by B, who therefore believes that Alice just sent him the key k_c she had, in fact, sent to Charlie in a former session. At the end of this execution, one can see that s is not secret anymore since Charlie knows k_c from his previous exchange and can perform a decryption of x_c to get s.

We can also illustrate Definition 4 using this example. Indeed, we have that for any s_1 and s_2, $P(s_1) \not\approx_l P(s_2)$. This is due to the fact that the two processes may evolve in two states that are not statically equivalent ($\approx_s$):

$$P(s_1) \overset{\alpha}{\rightarrow}{}^{*} P_1 = \nu(a, b, r_b).[A \mid \sigma_1] \quad \text{and} \quad P(s_2) \overset{\alpha}{\rightarrow}{}^{*} P_2 = \nu(a, b, r_b).[A \mid \sigma_2]$$

where $\alpha = c(M).\nu x_c.\bar{c}\langle x_c\rangle$. and σ_i is defined as σ where s is simply replaced by s_i. We can show that $P_1 \not\approx_s P_2$, due to the fact that the intruder can observe, using $N_1 = \mathsf{dec}(x_c, k_c)$ and $N_2 = s_1$, that $(N_1 =_E N_2)\sigma_1$ but $(N_1 \neq_E N_2)\sigma_2$.

3. Modeling the Norwegian protocol

We provide a formal specification of the Norwegian protocol, using the framework of the applied pi-calculus, defined in the previous section. We first model the cryptographic primitives used in the protocol (Section 3.1) and then the Norwegian protocol itself (Section 3.2).

3.1. Equational theory

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

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

The function ok is a constant; fst, hash, p, pk, s, snd, vk are unary functions; blind, d, dec, $+$, $*$, $\circ$, $\diamond$, pair, renc, sign, $\mathsf{unblind}$ are binary functions; $\mathsf{checkpfk}_1$, $\mathsf{checksign}$, penc are ternary functions; pfk_1, $\mathsf{checkpfk}_2$ are quaternary functions and pfk_2 is a quinary function.

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 $d_V(M)$ explained in Section 1.2. The term $\mathsf{blind}(M, N)$ represents the message M blinded by N. Unblinding 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 randomness N and public key P. The term $M \circ N$ denotes the homomorphic combination of ciphertexts and the corresponding operation is written $P \diamond Q$ on plaintexts and $R * S$ on random factors. The decryption of the 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. The addition of 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 $\mathsf{pfk}_1(M, N, P, Q)$ models a proof of knowledge, linked to the public identity M, that proves that Q is a ciphertext of the plaintext P using randomness N. It models what is provided to convince a prover, and its arguments represents the material needed to construct the proof itself. The term $\mathsf{pfk}_2(M, N, P, Q, R)$ represents a proof of knowledge linked to the public identity M, that R is a blinding of a re-encryption of a term Q using the secret key N and the blinding factor P. $\mathsf{pair}(M, N)$ represents the tuple (M, N). For simplicity, $\mathsf{pair}(M_1, \mathsf{pair}(\ldots, \mathsf{pair}(M_{n-1}, M_n)))$ may be abbreviated as $\langle M_1, \ldots, M_n\rangle$ and $\mathsf{fst}(\mathsf{snd}(M)^{i-1})$ as $\Pi_i(M)$ with $i \in \mathbb{N}$.

The properties of the primitives are then modeled by equipping the signature with an equational theory E that asserts that functions $+$, $*$, $\circ$ and $\diamond$ are commutative and associative, and includes the equations defined in Fig. 6. The first two equations are quite standard and models left and right projections of a pair of elements. The third equation simply represents usual decryption of a ciphertext using the corresponding secret key, while Eq. (E-4) reflects that a blinded ciphertext can be decrypted, yielding the corresponding blinded plaintext. Equation (E-5) models the homomorphic combination of ciphertexts. Equation (E-6) represents the re-encryption of a ciphertext. The operation of unblinding is described with Eq. (E-7). Equations (E-8), (E-9), and (E-10) correspond to the verification of respectively signature and proofs of knowledge pfk_1 and pfk_2.

The rewriting system corresponding to this equational theory is AC-convergent. This can be proved showing that the system is both AC-confluent and AC-terminating. The first property is true since there

$$
\begin{aligned}
\mathsf{fst}(\mathsf{pair}(x, y)) &= x && \text{(E-1)}\\
\mathsf{snd}(\mathsf{pair}(x, y)) &= y && \text{(E-2)}\\
\mathsf{dec}(\mathsf{penc}(x, r, \mathsf{pk}(k)), k) &= x && \text{(E-3)}\\
\mathsf{dec}(\mathsf{blind}(\mathsf{penc}(x, r, \mathsf{pk}(k)), b), k) &= \mathsf{blind}(x, b) && \text{(E-4)}\\
\mathsf{penc}(x, r_1, k_p) \circ \mathsf{penc}(y, r_2, k_p) &= \mathsf{penc}(x \diamond y, r_1 * r_2, k_p) && \text{(E-5)}\\
\mathsf{renc}(\mathsf{penc}(x, r, \mathsf{pk}(k_1)), k_2) &= \mathsf{penc}(x, r, \mathsf{pk}(k_1 + k_2)) && \text{(E-6)}\\
\mathsf{unblind}(\mathsf{blind}(x, b), b) &= x && \text{(E-7)}\\
\mathsf{checksign}(x, \mathsf{vk}(id), \mathsf{sign}(x, id)) &= \mathsf{ok} && \text{(E-8)}\\
\mathsf{checkpfk_1}(\mathsf{vk}(id), ball, \mathsf{pfk_1}(id, r, x, ball)) &= \mathsf{ok} && \text{(E-9)}\\
\text{where } ball &= \mathsf{penc}(x, r, k_p) &&\\
\mathsf{checkpfk_2}(\mathsf{vk}(id), x, ball, \mathsf{pfk_2}(\mathsf{vk}(id), k, b, x, ball)) &= \mathsf{ok} && \text{(E-10)}\\
\text{where } ball &= \mathsf{blind}(\mathsf{renc}(x, k), b) &&
\end{aligned}
$$

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

is no critical pairs. AC-termination can be shown through a special measure for length of terms $|\cdot|$ defined as follows:

$$
|M| = \begin{cases}
1 & \text{if } M \text{ is a name or a variable,}\\
2 + |M_1| + |M_2| + |M_3| & \text{if } M = \mathsf{penc}(M_1, M_2, M_3),\\
2 + |M_1| + |M_2| & \text{if } M = \mathsf{renc}(M_1, M_2),\\
1 + \sum_{i=1}^{k} |M_i| & \text{otherwise, i.e. } M = f(M_1, \dots, M_k).
\end{cases}
$$

Using this measure, it is easy to check that the length of terms is strictly decreasing at each step of the rewriting, which ensures AC-termination of the rewriting system.

3.2. Norwegian protocol process specification

We present here our model of the Norwegian voting protocol. Each player is modeled as an independent subprocess, and will be instantiated as a part of the whole protocol.

3.2.1. Voting process

The process $V(c_{bal}, c_{rec}, c_{pub}, k_{pub}, id_{vot}, pid_{rec}, v)$ represents both the voter and his computer.

$$
\begin{array}{ll}
\multicolumn{2}{l}{V(c_{bal}, c_{rec}, c_{pub}, k_{pub}, id_{vot}, pid_{rec}, v) = \nu\, r\ .}\\
\quad \text{let } e = \mathsf{penc}(v, r, k_{pub}),\ p = \mathsf{pfk_1}(id_{vot}, r, v, e),\ si = \mathsf{sign}(\langle e, p\rangle, id_{vot}) \text{ in} & \\
\quad \overline{c_{pub}}\langle\langle \mathsf{vk}(id_{vot}), e, p, si\rangle\rangle\ . & \%\ \text{Public information.}\\
\quad \overline{c_{bal}}\langle\langle \mathsf{vk}(id_{vot}), e, p, si\rangle\rangle\ . & \%\ \text{Encrypted ballot sent to } B.\\
\quad \overline{c_{rec}}\langle \mathsf{ok}\rangle\ . & \%\ \text{Synchronization for } R.\\
\quad c_{rec}(x_r)\ .\ c_{bal}(x_b)\ . & \%\ \text{Wait for inputs from } R \text{ and } B.\\
\multicolumn{2}{l}{\quad \text{if } \phi_{\mathsf{V}}(id_{vot}, pid_{rec}, v, e, p, si, x_b, x_r) \text{ then } \overline{c_{pub}}\langle \mathsf{ok}\rangle\ .\ \overline{c_{rec}}\langle \mathsf{ok}\rangle}
\end{array}
$$

with

$$\begin{aligned}\phi_{\mathsf{V}}(id_{vot}, pid_{rec}, v, e, p, si, x_b, x_r) = {}& \big(\mathsf{d}\big(\mathsf{p}(id_{vot}), \mathsf{blind}\big(v, \mathsf{s}(id_{vot})\big)\big) = x_r\big) \\ & \wedge \big(\mathsf{checksign}\big(\mathsf{hash}\big(\langle \mathsf{vk}(id_{vot}), e, p, si\rangle\big), pid_{rec}, x_b\big) = \mathsf{ok}\big).\end{aligned}$$

Parameter v represents voter's vote and c_{bal}, c_{rec} denote the authenticated channels shared with, respectively, the ballot box and the receipt generator. k_{pub} represents the public key of the election used to encrypt votes; id_{vot} is the secret id of the voter and pid_{rec} is the verification key of the receipt generator. Note that messages sent over c_{bal} and c_{rec} are also sent on the public channel c_{pub}. This simulates the fact that c_{bal} and c_{rec} are authenticated but not confidential channels. The synchronization step is only here to simplify the study in the case where B is corrupted. The formula $\phi_{\mathsf{V}}(id_{vot}, pid_{rec}, v, e, p, si, x_b, x_r)$ models all the checks performed by the voters: the message received from the ballot box should be properly signed and the message from the out-of-band channel should correspond to the right receipt code.

3.2.2. *Ballot box*

We represent the Ballot box, ready to listen to n voters, by the process B_n defined as follows.

$$\begin{array}{ll}
\multicolumn{2}{l}{B_n(c_{rec}, c_{dec}, c_{aud}, c_{pub}, k_{sec}, pid_{rec}, c^1_{vot}, pid^1_{vot}, s^1_{vot}, \ldots, c^n_{vot}, pid^n_{vot}, s^n_{vot}) =} \\
\quad c^1_{vot}(x_1)\ .\ BB_1\ .\ \ldots\ .\ c^n_{vot}(x_n)\ .\ BB_n\ . & \%\ \text{Processes incoming votes.} \\
\quad \overline{c_{bd}}\langle \Pi_2(x_1)\rangle\ .\ \ldots\ .\ \overline{c_{bd}}\langle \Pi_2(x_n)\rangle\ . & \%\ \text{Outputs encrypted votes to } D. \\
\quad \overline{c_{ba}}\langle x_1\rangle\ .\ \ldots\ .\ \overline{c_{ba}}\langle x_n\rangle & \%\ \text{Outputs content to } A.
\end{array}$$

with:

$$\begin{array}{ll}
BB_i = & \\
\quad \text{if } \phi_{\mathsf{B}}(pid^i_{vot}, x_i) \text{ then} & \%\ \text{Checks ballot's validity.} \\
\quad \text{let } b_i = \mathsf{blind}(\mathsf{renc}(\Pi_2(x_i), k_{sec}), s^i_{vot}) \text{ in} & \%\ \text{Computes re-encrypted blinded} \\
\quad \text{let } pok_i = \mathsf{pfk}_2(pid^i_{vot}, k_{sec}, s^i_{vot}, \Pi_2(x_i), b_i) \text{ in} & \quad\ \text{ballot and corresponding proof.} \\
\quad \overline{c_{rec}}\langle\langle x_i, b_i, pok_i\rangle\rangle\ .\ c_{rec}(y_i)\ . & \%\ \text{Message sent to } R.\ \text{Wait for } R. \\
\quad \text{if } \phi_{\mathsf{S}}(pid_{rec}, x_i, y_i) \text{ then } \overline{c_{pub}}\langle y_i\rangle\ .\ \overline{c^i_{vot}}\langle y_i\rangle & \%\ \text{Checks confirmation's validity and} \\
 & \quad\ \text{sends confirmation to the Voter.}
\end{array}$$

and

$$\begin{aligned}
\phi_{\mathsf{S}}(M, N, U) &= \big(\mathsf{checksign}\big(\mathsf{hash}(N), M, U\big) = \mathsf{ok}\big), \\
\phi_{\mathsf{B}}(V, W) &= \big(W = \langle W_1, W_2, W_3, W_4\rangle\big) \wedge \big(\mathsf{checkpfk}_1(W_1, W_2, W_3) = \mathsf{ok}\big) \\
&\qquad \wedge (W_1 = V) \wedge \big(\mathsf{checksign}\big(\langle W_2, W_3\rangle, W_1, W_4\big) = \mathsf{ok}\big)
\end{aligned}$$

where $(X = \langle X_1, \ldots, X_n\rangle)$ denotes the formula that holds only when X is a n-tuple. For example, $(X = \langle X_1, X_2\rangle)$ denotes the formula $X = \mathsf{pair}(\mathsf{fst}(X), \mathsf{snd}(X))$.

Intuitively, we assume the ballots to be received from the authenticated channels $c^1_{vot}, \ldots, c^n_{vot}$. The Ballot box processes each ballot one after another, exchanging with the Receipt generator through the secure channel c_{rec}, before sending back a confirmation to the Voter. Once all votes have been casted, the Ballot box outputs the encrypted votes to the Decryption device using the secure channel c_{dec}, and its content to the Auditor through the secure channel c_{aud}. k_{sec} is the secret key known by B, while

$pid^1_{vot}, \dots, pid^n_{vot}$ are the public identities of the voters (i.e. their verification keys) and $s^1_{vot}, \dots, s^n_{vot}$ the corresponding blinding factors.

3.2.3. *Receipt generator*

$R_n(c_{bal}, c_{aud}, c_{dec}, c_{pub}, k_{sec}, id_{rec}, c^1_{vot}, pid^1_{vot}, p^1_{vot}, \dots, c^n_{vot}, pid^n_{vot}, p^n_{vot})$ is the Receipt generator's process. It exchanges messages with the Ballot box, the Decryption service (only for an ad-hoc synchronization, used to simplify the proof) and the Auditor through secure channels c_{bal}, c_{dec}, and c_{aud} respectively. It also talks directly to voters through out-of-band channels $c^1_{vot}, \dots, c^n_{vot}$, which are modeled as authenticated channels here. k_{sec} is the secret key received by R during the setup phase, $pid^1_{vot}, \dots, pid^n_{vot}$ are the public identities of the voters and the corresponding receipt coding functions are $p^1_{vot}, \dots, p^n_{vot}$.

$$
\begin{array}{ll}
\multicolumn{2}{l}{R_n(c_{bal}, c_{aud}, c_{dec}, c_{pub}, k_{sec}, id_{rec}, c^1_{vot}, pid^1_{vot}, p^1_{vot}, \dots, c^n_{vot}, pid^n_{vot}, p^n_{vot}) =} \\
\quad c^1_{vot}(sync^1_r)\ .\ RG_1\ .\ c^1_{vot}(sync^1_v)\ . & \text{\% Processes re-encrypted votes into receipts.} \\
\quad \dots & \\
\quad c^n_{vot}(sync^n_r)\ .\ RG_n\ .\ c^n_{vot}(sync^n_v)\ . & \\
\quad \overline{c_{aud}}\langle\langle pid^1_{vot}, hbr_1\rangle\rangle\ .\ \dots\ .\ \overline{c_{aud}}\langle\langle pid^n_{vot}, hbr_n\rangle\rangle\ . & \text{\% Outputs content to } A. \\
\quad \overline{c_{dec}}\langle \mathsf{ok}\rangle & \text{\% Ad-hoc synchronization for } D.
\end{array}
$$

with:

$$
\begin{array}{ll}
\multicolumn{2}{l}{RG_i =} \\
\quad c_{bal}(x_i)\ . & \text{\% Waiting input from the Ballot box.} \\
\quad \text{if } \phi_{\mathsf{R}}(ipid^i_{vot}, x_i) \text{ then} & \text{\% Checks Ballot box's computations.} \\
\quad \text{let } r_i = \mathsf{d}(p^i_{vot}, \mathsf{dec}(\Pi_2(x_i), k_{sec})) \text{ in} & \text{\% Computes receipt for the i-th Voter.} \\
\quad \text{let } hbr_i = \mathsf{hash}(\Pi_1(x_i)) \text{ in} & \\
\quad \text{let } si_i = \mathsf{sign}(hbr_i, id_{rec}) \text{ in} & \text{\% Computes i-th confirmation for the Ballot box.} \\
\quad \overline{c_{bal}}\langle si_i\rangle\ .\ \overline{c_{pub}}\langle r_i\rangle\ .\ \overline{c^i_{vot}}\langle r_i\rangle & \text{\% Outputs to intended recipients.}
\end{array}
$$

and

$$
\begin{aligned}
\phi_{\mathsf{R}}(X, Y) = {} & \big(Y = \langle Y_1, Y_2, Y_3\rangle\big) \wedge \big(Y_1 = \langle W_1, W_2, W_3, W_4\rangle\big) \wedge (W_1 = X) \\
& \wedge \big(\mathsf{checksign}\big(\langle W_2, W_3\rangle, W_1, W_4\big) = \mathsf{ok}\big) \wedge \big(\mathsf{checkpfk_1}(W_1, W_2, W_3) = \mathsf{ok}\big) \\
& \wedge \big(\mathsf{checkpfk_2}(W_1, W_2, Y_2, Y_3) = \mathsf{ok}\big).
\end{aligned}
$$

Note that instructions $c^i_{vot}(sync^i_r)$ and $c^i_{vot}(sync^i_v)$ are used to force the Receipt Generator to fully process each receipt before accepting a new entry from the Ballot box, which eases the security analysis. Note also that we slightly simplify the behavior of the Receipt Generator. [23] indicates that the Receipt Generator not only computes hbr_i as defined above but also the hash of the voter's ballot (obtained when computing $\Pi_1(x_i)$) from which the signature is dropped. We ignore the second hash as it contains less information.

3.2.4. *Decryption service*

The Decryption service is represented by the process $D_n(c_{bal}, c_{rec}, c_{aud}, c_{pub}, k_{sec})$. It communicates securely with the Ballot box, the Receipt generator (waiting for synchronization), and the Auditor through respectively channels c_{bal}, c_{rec}, and c_{aud}. The result is published on the public channel c_{pub}. In order to decrypt ballots, it needs to know the secret key k_{sec}. The parallelism at the end of the process models that the votes are shuffled. For simplicity, we omit the proof of correct decryption provided by the Decryption device as it should not affect privacy.

$$\begin{array}{ll}
D_n(c_{bal}, c_{rec}, c_{aud}, c_{pub}, k_{sec}) = & \\
\quad c_{rec}(sync_d^1)\ . & \text{\% Waits for } R\text{'s signal to begin the tallying phase.} \\
\quad c_{bal}(d_1)\ .\ \dots\ .\ c_{bal}(d_n)\ . & \text{\% Inputs encrypted votes from } B. \\
\quad \overline{c_{aud}}\langle \mathsf{hash}(\langle d_1, \dots, d_n\rangle)\rangle\ . & \text{\% Outputs hashed tally for } A. \\
\quad c_{aud}(sync_d^2)\ . & \text{\% Waits } A \text{ approval before processing the outcome.} \\
\quad (\overline{c_{pub}}\langle \mathsf{dec}(d_1, k_{sec})\rangle \mid \cdots \mid \overline{c_{pub}}\langle \mathsf{dec}(d_n, k_{sec})\rangle) & \text{\% Publishes the results in a non-deterministic order.}
\end{array}$$

3.2.5. Auditor

Finally, the Auditor is modeled by the process $A_n(c_{bal}, c_{rec}, c_{dec})$ which communicates with the other infrastructure players (Ballot box, Receipt generator, and Decryption device) using secure channels c_{bal}, c_{rec}, and c_{dec}.

$$\begin{array}{ll}
A_n(c_{bal}, c_{rec}, c_{dec}) = & \\
\quad c_{rec}(h_1)\ .\ \dots\ .\ c_{rec}(h_n)\ . & \text{\% Inputs from } R,\ D \text{ and } B. \\
\quad c_{dec}(h_d)\ .\ c_{bal}(x_1)\ .\ \dots\ .\ c_{bal}(x_n)\ . & \\
\quad \text{if } \phi_{\mathsf{A}}(h_d, h_1, \dots, h_n, x_1, \dots, x_n) \text{ then } \overline{c_{dec}}\langle \mathsf{ok}\rangle \text{ else } 0 & \text{\% Checks and sends approval.}
\end{array}$$

with:

$$\begin{aligned}
\phi_{\mathsf{A}}(H, X_1, \dots, X_n, Y_1, \dots, Y_n) = {} & \big(\mathsf{hash}\big(\langle \Pi_2(Y_1), \dots, \Pi_2(Y_n)\rangle\big) = H\big) \\
& \bigwedge_{i=1}^{n}\big[\big(Y_i = \langle W_1, W_2, W_3, W_4\rangle\big) \wedge \big(X_i = \langle Z_1, Z_2\rangle\big) \wedge (W_1 = Z_1) \\
& \wedge \big(\mathsf{hash}(Y_i) = Z_2\big) \wedge \big(\mathsf{checksign}\big(\langle W_2, W_3\rangle, W_1, W_4\big) = \mathsf{ok}\big)\big].
\end{aligned}$$

3.2.6. Norwegian protocol and corruption scenarios

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 names a_1, a_2, a_3 model the private keys used in the protocol and the corresponding public keys $\mathsf{pk}(\mathsf{a}_1)$, $\mathsf{pk}(\mathsf{a}_2)$ and $\mathsf{pk}(\mathsf{a}_3)$ are added in the process frame. The restricted names c_1, c_2 (resp. $\mathsf{c}_{\mathsf{RV}_1}$ and $\mathsf{c}_{\mathsf{RV}_2}$) model authentic channels between the two honest voters and the Ballot box (resp. the Receipt generator). The restricted names id_1, id_2, id_R represent the secret ids of honest voters and of the Receipt generator. The corresponding public id's are added to the process frame.

The process corresponding to the situation where all the authorities are honest is $P_n[\_]$ where n is the number of voters and the hole is the voter's place and is defined as follows:

$$\begin{aligned}
P_n[\_] = \nu\tilde{n}.(\text{ let } \mathsf{a}_3 = \mathsf{a}_1 + \mathsf{a}_2 \text{ in }).\big[\ \_ \\
& \mid B_n\big(\mathsf{c}_{\mathsf{BR}}, \mathsf{c}_{\mathsf{BD}}, \mathsf{c}_{\mathsf{BA}}, \mathsf{c}_{\mathsf{out}}, \mathsf{a}_2, \mathsf{idp}_\mathsf{R}, \mathsf{c}_1, \mathsf{idp}_1, \mathsf{s}(\mathsf{id}_1), \dots, \mathsf{c}_\mathsf{n}, \mathsf{idp}_\mathsf{n}, \mathsf{s}(\mathsf{id}_\mathsf{n})\big) \\
& \mid R_n\big(\mathsf{c}_{\mathsf{BR}}, \mathsf{c}_{\mathsf{RA}}, \mathsf{c}_{\mathsf{RD}}, \mathsf{c}_{\mathsf{out}}, \mathsf{a}_3, \mathsf{id}_\mathsf{R}, \mathsf{c}_{\mathsf{RV}_1}, \mathsf{idp}_1, \mathsf{p}(\mathsf{id}_1), \dots, \mathsf{c}_{\mathsf{RV}_\mathsf{n}}, \mathsf{idp}_\mathsf{n}, \mathsf{p}(\mathsf{id}_\mathsf{n})\big) \\
& \mid D_n(\mathsf{c}_{\mathsf{BD}}, \mathsf{c}_{\mathsf{RD}}, \mathsf{c}_{\mathsf{DA}}, \mathsf{c}_{\mathsf{out}}, \mathsf{a}_1) \mid A_n(\mathsf{c}_{\mathsf{BA}}, \mathsf{c}_{\mathsf{RA}}, \mathsf{c}_{\mathsf{DA}}) \mid \Gamma\big]
\end{aligned}$$

with $\tilde{n} = \mathsf{a_1, a_2, a_3, id_1, id_2, r_1, r_2, id_R, c_1, c_2, c_{RV_1}, c_{RV_2}, c_{BA}, c_{RA}, c_{DA}, c_{BR}, c_{BD}, c_{RD}}$ the set of restricted names and the frame $\Gamma = \{^{\mathsf{pk(a_i)}}/_{\mathsf{g_i}} \mid \mathsf{i} = 1, \ldots, 3\} \mid \{^{\mathsf{vk(id_i)}}/_{\mathsf{idp_i}} \mid \mathsf{i} = 1, 2\} \mid \{^{\mathsf{vk(id_R)}}/_{\mathsf{idp_R}}\}$. This frame 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.

The process P_n corresponds therefore to a scenario where all the election authorities are honest. To model the case where the Ballot box is corrupted, we simply provide the attacker with the Ballot box's secrets. Formally, we define the process P_n^b defined as follows.

$$\begin{aligned} P_n^b[\_] = \nu\tilde{n}_b.(\text{ let } \mathsf{a_3} = \mathsf{a_1} + \mathsf{a_2} \text{ in }).\big[\ \_ \\ &\mid R_n\big(\mathsf{c_{BR}, c_{RA}, c_{RD}, c_{out}, a_3, id_R, c_{RV_1}, idp_1, p(id_1)}, \ldots, \mathsf{c_{RV_n}, idp_n, p(id_n)}\big) \\ &\mid D_n(\mathsf{c_{BD}, c_{RD}, c_{DA}, c_{out}, a_1}) \mid A_n(\mathsf{c_{BA}, c_{RA}, c_{DA}}) \mid \Gamma_b\big] \end{aligned}$$

with $\tilde{n}_b = \mathsf{a_1, a_3, id_1, id_2, r_1, r_2, id_R, c_{RV_1}, c_{RV_2}, c_{RA}, c_{DA}, c_{RD}}$ the set of restricted names and $\Gamma_b = \{^{\mathsf{pk(a_i)}}/_{\mathsf{g_i}} \mid \mathsf{i} = 1, \ldots, 3\} \mid \{^{\mathsf{vk(id_i)}}/_{\mathsf{idp_i}}, {}^{\mathsf{s(id_i)}}/_{\mathsf{s_i}} \mid \mathsf{i} = 1, 2\} \mid \{^{\mathsf{vk(id_R)}}/_{\mathsf{idp_R}}\}$. Compared to P_n, we have simply removed B_n from the process (since the Ballot box is now under the control of the attacker) and the secret key $\mathsf{a_2}$ and the authenticated channels $\mathsf{c_1, c_2, c_{BA}, c_{BR}, c_{BD}}$ are now public (they are not part of the set of restricted names anymore). Finally, we have added $\mathsf{s(id_1), s(id_2)}$ to the frame representing the initial knowledge. This models the fact that now, all the secrets of B_n are known to the attacker.

4. Formal analysis of ballot secrecy

Our analysis shows that the Norwegian e-voting protocol preserves ballot secrecy, even when the Ballot box and all but two voters are corrupted, provided that the other components are honest. This of course implies ballot secrecy if all the authorities are honest and some voters are corrupted. Conversely, we identified several cases of corruption that are subject to attacks. Though not surprising, these cases were not explicitly mentioned in the literature.

In this section, we state our main security results, studying the privacy of the Norwegian protocol under two main corruptions scenarios. Moreover, we summarize existing attack scenarios. The formal proof of privacy is then detailed in the next three sections.

Ballot secrecy has been formally defined in terms of equivalence by Delaune, Kremer, and Ryan in [21]. A protocol with 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:

$$\nu\tilde{n}.\big(A \mid V(v_1, a_1) \mid V(v_2, a_2)\big) \approx_l \nu\tilde{n}.\big(A \mid V(v_2, a_1) \mid V(v_1, a_2)\big)$$

Proving ballot secrecy of the Norwegian protocol therefore amounts in proving equivalence of the corresponding processes we detailed in Section 3.

4.1. Corrupted ballot box and corrupted voters

Our main result states that the Norwegian protocol specification satisfies ballot secrecy even if the Ballot box and $n-2$ voters are corrupted, provided that the other components are honest.

Theorem 1. *Let n be an integer representing the number of voters. Let P_n^b be the process defined in Section 3.2.6, that corresponds to the voting process where all the authorities but the Ballot box are honest. Then,*

$$P_n^b\big[V_1(\mathsf{a}) \mid V_2(\mathsf{b})\big] \approx_l P_n^b\big[V_1(\mathsf{b}) \mid V_2(\mathsf{a})\big]$$

with $V_i(\mathsf{v_j}) = V(\mathsf{c_i}, \mathsf{c_{RV_i}}, \mathsf{c_{out}}, \mathsf{g_1}, \mathsf{id_i}, \mathsf{idp_R}, \mathsf{v_j})$ *which corresponds to the i-th voter voting* $\mathsf{v_j}$.

In order to prove Theorem 1, we need to "guess" a symmetric relation $\mathcal{R}$ on closed processes satisfying the properties of a labelled bisimilarity. This amounts into describing symbolically all the possible (co-)evolutions of the two processes, depending on the actions of the adversary. The description of this relation is given in Section F. The first step of the proof consists in showing $\mathcal{R}$ satisfies property (1) of a bisimilarity relation, that is, we need to show that all pairs of frames obtained in the relation $\mathcal{R}$ are in static equivalence. In fact, all these frames are included in the final frames (modulo a "cleaning" step performed using Lemma 12 presented in appendix) corresponding to the complete execution of the two processes. It is therefore sufficient to prove static equivalence of the two final frames.

We consider the following frames:

$$\begin{aligned}
\theta_{\mathsf{init}} &= \big\{{}^{\mathsf{vk(id_k)}}/_{\mathsf{idp_k}}, {}^{\mathsf{s(id_k)}}/_{\mathsf{s_k}} \mid \mathsf{k} = 1, \dots, n\big\} \mid \big\{{}^{\mathsf{vk(id_R)}}/_{\mathsf{idp_R}}\big\} \mid \big\{{}^{\mathsf{pk(a_k)}}/_{\mathsf{g_k}} \mid \mathsf{k} = 1, \dots, 3\big\},\\
\theta_0 &= \theta_{\mathsf{init}} \mid \big\{{}^{\mathsf{penc(v_k,r_k,g_1)}}/_{\mathsf{e_k}}, {}^{\mathsf{pfk_1(id_k,t_k,v_k,e_k)}}/_{\mathsf{p_k}}, {}^{\mathsf{sign(\langle e_k,p_k\rangle,id_k)}}/_{\mathsf{si_k}} \mid \mathsf{k} = 1, 2\big\},\\
\theta_k &= \theta_{k-1} \mid \big\{{}^{\mathsf{sign(hash(}\Pi_1(M_\mathsf{k})\mathsf{),id_R)}}/_{sr_k}, {}^{\mathsf{d(p(id_k),dec(}\Pi_2(M_\mathsf{k})\mathsf{,a_3))}}/_{rec_k}\big\},\\
\theta_\delta &= \theta_n \mid \big\{{}^{\mathsf{dec(}U_{\delta(k)}\mathsf{,a_1)}}/_{res_k} \mid \mathsf{k} = 1, \dots, n\big\},
\end{aligned}$$

where M_i and U_k are free terms such that $\mathsf{fv}(M_{i+1}) \subseteq \mathsf{dom}(\theta_i)$ and $\mathsf{fv}(U_{k+1}) \subseteq \mathsf{dom}(\theta_n)$. Intuitively, the M_i and U_k are the recipes sent by the adversary. The restriction on the variables makes sure the adversary only use terms he has access to, at this step. δ is a substitution of $[\![1, n]\!]$ intuitively corresponding to the shuffling of the votes at the end of the election. Then each frame can be interpreted as follows:

- θ_{init} corresponds to the initial knowledge of the attacker. It contains the public data of the honest voters and the public keys of the election.
- θ_0 corresponds to the submission of ballots from the two honest voters. Note that our synchronization phase ensures that honest voters vote first. And we can show that the adversary cannot interfere with these two ballots.
- θ_k corresponds to the knowledge of the adversary once the k-th voter has voted. Intuitively, the adversary will submit any ballot he wishes (M_k) based on his prior knowledge and in return, he receives the receipt and the signature from the Receipt generator, that is, he receives $\mathsf{d(p(id_k), dec(}\Pi_2(M_\mathsf{k})\mathsf{, a_3))}$ and $\mathsf{sign(hash(}\Pi_1(M_\mathsf{k})\mathsf{), id_R)}$.
- Then θ_δ corresponds to the frame with the final decryption of the votes, after shuffling, that is, the adversary can see the votes in clear after some permutation δ.

Proposition 1. *Let δ is a substitution of $[\![1,n]\!]$ and ${}^t\delta = \delta \circ [1 \mapsto 2, 2 \mapsto 1]$. Let θ_δ be the frame as defined above. Then we have:*

$$\nu\tilde{\omega}.\theta_\delta\sigma_{\mathsf{L}} \approx_s \nu\tilde{\omega}.\theta_{{}^t\delta}\sigma_{\mathsf{R}}, \quad \textit{with } \sigma_{\mathsf{L}} = \left\{{}^{\mathsf{a}}/_{\mathsf{v_1}}, {}^{\mathsf{b}}/_{\mathsf{v_2}}\right\} \textit{ and } \sigma_{\mathsf{R}} = \left\{{}^{\mathsf{b}}/_{\mathsf{v_1}}, {}^{\mathsf{a}}/_{\mathsf{v_2}}\right\}.$$

This proposition is the main core result to establish Theorem 1.

Proof sketch. The proof is done step by step. First, we show that $\nu\tilde{\omega}.\theta_0\sigma_L \approx_s \nu\tilde{\omega}.\theta_0\sigma_R$, that is, the frames are in static equivalence once the two honest voters have voted (Lemma 13 detailed in appendix). We then show that the receipt sent by the receipt generator does not break static equivalence. This requires to prove in particular that adding the signature of a known term does preserve static equivalence (Lemma 8, one of our core lemma). Finally, we conclude by showing that the decryption of the (shuffled) vote only yields already known terms, built using the same recipes, in both frames.

The full proof of Proposition 1 can be found in Section 6. □

It then remains to show that we did not miss any possible execution, that is, show that $\mathcal{R}$ is a bisimilarity relation, which is ensured by the following proposition.

Proposition 2. *Let $\mathcal{R}$ be the relation defined in Definition 15 (Section F). Then, $\mathcal{R}$ is verifying properties* (2) *and* (3) *of Definition 4.*

Proof sketch. Given $(P, Q) \in \mathcal{R}$, we consider all possible evolutions of P in P' and show that there exists Q' such that (P', Q') remains in $\mathcal{R}$. In other words, we check that we did not forget any case when defining $\mathcal{R}$. The detailed proof is not really technical but is rather tedious and is therefore deferred to Section F. □

Theorem 1 then easily follows from Proposition 1 and Proposition 2.

4.2. Honest authorities and corrupted voters

The Norwegian e-Voting Protocol specification *a fortiori* satisfies ballot secrecy even if $n-2$ voters are corrupted, provided that the other components are honest.

Theorem 2. *Let n be an integer, that corresponds to the number of voters. Let P_n be the process defined in Section 3.2.6, that corresponds to the voting process when all authorities are honest. Then,*

$$P_n\big[V_1(\mathsf{a}) \mid V_2(\mathsf{b})\big] \approx_l P_n\big[V_1(\mathsf{b}) \mid V_2(\mathsf{a})\big]$$

with $V_i(\mathsf{v_j}) = V(\mathsf{c_i}, \mathsf{c_{RV_i}}, \mathsf{c_{out}}, \mathsf{g_1}, \mathsf{id_i}, \mathsf{idp_R}, \mathsf{v_j})$ *which corresponds to the i-th voter voting* $\mathsf{v_j}$.

Theorem 2 is a corollary of Theorem 1. While this is intuitively obvious, the use of equivalence adds some technicalities to the proof. The key proposition is that extending the initial knowledge of the attacker can only help finding attacks.

Lemma 1. *Let $\tilde{n}$ be a set of names, P, Q, two processes. Then, we have:*

$$\nu\tilde{n}.\big(P \mid \{{}^M/_x\}\big) \approx_l \nu\tilde{n}.\big(Q \mid \{{}^M/_x\}\big) \quad \Longrightarrow \quad \nu\tilde{n}.P \approx_l \nu\tilde{n}.Q.$$

Proof. We define a symmetric relation $\mathcal{R}$ on closed extended processes as follows:

$$\nu\tilde{n}.A\,\mathcal{R}\,\nu\tilde{n}.B \quad \overset{\text{def}}{\Longleftrightarrow} \quad \nu\tilde{n}.\big(A \mid \{^M/_x\}\big) \approx_l \nu\tilde{n}.\big(B \mid \{^M/_x\}\big).$$

Let us show that $\mathcal{R}$ verifies the three properties of a bisimilarity relation (*cf* Definition 4).

1. This is straightforward since $\phi(\nu\tilde{n}.A) \subseteq \phi(\nu\tilde{n}.(A \mid \{^M/_x\}))$.
2. Let $A \rightarrow A'$. Then $\nu\tilde{n}.(A \mid \{^M/_x\}) \rightarrow \nu\tilde{n}.(A' \mid \{^M/_x\})$. Now, since $A\,\mathcal{R}\,B$, we have that $\nu\tilde{n}.(A \mid \{^M/_x\}) \approx_l \nu\tilde{n}.(B \mid \{^M/_x\})$ thus there exists $\overline{B}$ such that $\nu\tilde{n}.(B \mid \{^M/_x\}) \rightarrow^* \overline{B}$ and $\overline{B} \approx_l \nu\tilde{n}.(A' \mid \{^M/_x\})$. Since B is a closed process, it can not make use of x thus it must be the case that there exists B' such that $B \rightarrow^* B'$ and $\overline{B} \equiv \nu\tilde{n}.(B' \mid \{^M/_x\})$. Moreover since $\nu\tilde{n}.(A' \mid \{^M/_x\}) \approx_l \overline{B} \equiv \nu\tilde{n}.(B' \mid \{^M/_x\})$, we conclude that $A'\,\mathcal{R}\,B'$.
3. Let $A \xrightarrow{\alpha} A'$. Then $\nu\tilde{n}.(A \mid \{^M/_x\}) \xrightarrow{\alpha} \nu\tilde{n}.(A' \mid \{^M/_x\})$. Now, since $A\,\mathcal{R}\,B$, we have that $\nu\tilde{n}.(A \mid \{^M/_x\}) \approx_l \nu\tilde{n}.(B \mid \{^M/_x\})$ thus there exists $\overline{B}$ such that $\nu\tilde{n}.(B \mid \{^M/_x\}) \rightarrow^*\xrightarrow{\alpha}\rightarrow^* \overline{B}$ and $\overline{B} \approx_l \nu\tilde{n}.(A' \mid \{^M/_x\})$. Since A is a closed process, it does not contain x, thus we must have that the label α does not contain x either. Thus, if $\nu\tilde{n}.(B \mid \{^M/_x\}) \rightarrow^*\xrightarrow{\alpha}\rightarrow^* \overline{B}$, we must have that there exists B' such that $B \rightarrow^*\xrightarrow{\alpha}\rightarrow^* B'$ and $\overline{B} \equiv \nu\tilde{n}.(B' \mid \{^M/_x\})$. Moreover since $\nu\tilde{n}.(A' \mid \{^M/_x\}) \approx_l \overline{B} \equiv \nu\tilde{n}.(B' \mid \{^M/_x\})$, we conclude that $A'\,\mathcal{R}\,B'$.

Now, since $\approx_l$ is the largest relation satisfying these three properties, we conclude. □

We are now ready to prove Theorem 2.

Proof of Theorem 2. Let us define $A = P_n^b[V_1(\mathsf{a}) \mid V_2(\mathsf{b})]$ and $B = P_n^b[V_1(\mathsf{b}) \mid V_2(\mathsf{a})]$. According to Theorem 1, we have that: $A \approx_l B$. Thus, for all closing evaluation context $C[\_]$, we also have: $C[A] \approx_l C[B]$. Let us consider the following closing evaluation context:

$$C[\_] = \nu\tilde{m}.\big[\,\_ \mid B_n(\mathsf{c_{BR}}, \mathsf{c_{BD}}, \mathsf{c_{BA}}, \mathsf{c_{out}}, \mathsf{a_2}, \mathsf{idp_R}, \mathsf{c_1}, \mathsf{idp_1}, \mathsf{s_1}, \dots, \mathsf{c_n}, \mathsf{idp_n}, \mathsf{s_n})\big],$$

with $\tilde{m} = \mathsf{a_2}, \mathsf{c_1}, \mathsf{c_2}, \mathsf{c_{BR}}, \mathsf{c_{BD}}$ and B_n is defined as the honest one except that we replace arguments $\mathsf{s(id_1)}$ and $\mathsf{s(id_2)}$ by variables $\mathsf{s_1}$ and $\mathsf{s_2}$. Then:

$$C[A] = \nu\tilde{m}.\big[\nu\tilde{n}_b.(\text{ let } \mathsf{a_3} = \mathsf{a_1} + \mathsf{a_2} \text{ in }).\big(V_1(\mathsf{a}) \mid V_2(\mathsf{b}) \mid R_n \mid D_n \mid \Gamma_b\big)\sigma_L \mid B_n\big]$$

where B_n, R_n and D_n are just short notations of the different processes with their attributes. Since B_n is such that $(\mathsf{fv}(B_n) \cup \mathsf{fn}(B_n)) \cap \tilde{n}_b = \emptyset$, we have that:

$$\begin{aligned} C[A] &= \nu\tilde{m}.\big[\nu\tilde{n}_b.(\text{ let } \mathsf{a_3} = \mathsf{a_1} + \mathsf{a_2} \text{ in }).\big(V_1(\mathsf{a}) \mid V_2(\mathsf{b}) \mid B_n \mid R_n \mid D_n \mid \Gamma_b\big)\big] \\ &= \nu(\tilde{m}, \tilde{n}_b).(\text{ let } \mathsf{a_3} = \mathsf{a_1} + \mathsf{a_2} \text{ in }).\big[V_1(\mathsf{a}) \mid V_2(\mathsf{b}) \mid B_n \mid R_n \mid D_n \mid \Gamma_b\big]. \end{aligned}$$

But $(\tilde{m}, \tilde{n}_b) = \tilde{n}$ and:

$$\begin{aligned} & B_n(\mathsf{c_{BR}}, \mathsf{c_{BD}}, \mathsf{c_{BA}}, \mathsf{c_{out}}, \mathsf{a_2}, \mathsf{idp_R}, \mathsf{c_1}, \mathsf{idp_1}, \mathsf{s_1}, \dots, \mathsf{c_n}, \mathsf{idp_n}, \mathsf{s_n})\Gamma_b \\ &\quad = B_n\big(\mathsf{c_{BR}}, \mathsf{c_{BD}}, \mathsf{c_{BA}}, \mathsf{c_{out}}, \mathsf{a_2}, \mathsf{idp_R}, \mathsf{c_1}, \mathsf{idp_1}, \mathsf{s(id_1)}, \dots, \mathsf{c_n}, \mathsf{idp_n}, \mathsf{s(id_n)}\big)\Gamma_b. \end{aligned}$$

So, we get that $C[A] = \overline{P}_n[V_1(\mathsf{a}) \mid V_2(\mathsf{b})]$, where $\overline{P}_n^a$ is exactly the same as P_n except that Γ is replaced by Γ_b. Doing the same for $C[B]$, we deduce that $C[B] = \overline{P}_n[V_1(\mathsf{b}) \mid V_2(\mathsf{a})]$ and thus $\overline{P}_n[V_1(\mathsf{a}) \mid V_2(\mathsf{b})] \approx_l \overline{P}_n[V_1(\mathsf{b}) \mid V_2(\mathsf{a})]$. According to Lemma 1, this equivalence implies that:

$$P_n\big[V_1(\mathsf{a}) \mid V_2(\mathsf{b})\big] \approx_l P_n\big[V_1(\mathsf{b}) \mid V_2(\mathsf{a})\big]. \qquad \square$$

4.3. *Attacks*

We have shown that the Norwegian protocol guarantees ballot privacy provided that the Receipt generator, the Decryption service, and the Auditor are honest. We review further 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 Ballot box. Such an attack is not possible in [23,24] since communications are assumed to be secure (e.g. using TLS channels). It is interesting to note that the combination of both studies indicates that in case the Decryption service is corrupted then ballot secrecy solely relies on the security of the communication channels between the voters and the Ballot box.

Dishonest ballot box and receipt generator. Clearly, if the Ballot box and the Receipt generator collude, they can compute $\mathsf{a}_1 = \mathsf{a}_3 - \mathsf{a}_2$ and they can then decrypt all incoming encrypted ballots. More interestingly, a corrupted Receipt generator does not need the full cooperation of the ballot 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 Ballot box to blind the ballot. Recall that the Receipt generator retrieves $f(o)^{s_V}$ when generating the receipt codes (by computing $\check{\mathsf{w}}\check{\mathsf{x}}^{-\mathsf{a}_3}$). Therefore, the Receipt generator can compute $\mathsf{f}(\mathsf{o}')^{\mathsf{s}_\mathsf{V}}$ for any possible voting option o'. Comparing with the obtained values with $\mathsf{f}(\mathsf{o})^{\mathsf{s}_\mathsf{V}}$ 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 (who has access to the data during the generation? How the data are distributed and how secure the distribution is?). The Ballot box can also perform a similar attack provided that it has access to the SMS sent by the Receipt generator and provided it can learn the coding function d_V for some voters V (which probably requires partial corruption of the Receipt generator as well). Note that if the Ballot box and the Receipt generator collude, verifiability is broken as well, as pointed in [23,24], since votes can be changed without voters noticing.

Dishonest ballot box and auditor. Even if the Auditor does not hold any secret (besides the access to the output of both the Ballot box and the Receipt generator), it is still a key component in the voting process. Indeed, it ensures that the ballots sent to the Decryption service indeed correspond to the ballots sent by the voters (unless both the Ballot box and the Receipt generator are corrupted). Assume now that the Ballot box and the Auditor corrupted. Then the Ballot box can send any ballots it wants to the Decryption service (the – corrupted – Auditor would not complain). This is a clear breach of security in terms of the correctness of the result: indeed, the results would not correspond to the votes as casted by the voters (as also pointed in [23,24]). As a consequence, this is also a breach of ballot privacy. Indeed, the Ballot box may send the same ballot several times (possibly re-randomized) therefore obtaining a bias of information about the vote casted by the voter under attack.

5. Lemmas for static equivalence

The proof of our main result requires several intermediate lemmas. We believe that some of them are of independent of interest. We first state lemmas that are independent of the equational theory (Section 5.1) and then one of them that relies on it but is still quite general and could be reused for other formal studies of protocols (Section 5.2).

5.1. Generic lemmas

We use some arguments repetitively on our proofs and we found useful to state them separately. First, we notice that splitting pairs preserves static equivalence.

Lemma 2. *Let $\tilde{n}, \tilde{m}$ be names, θ_1, θ_2 substitutions and let us define $\phi = \nu\tilde{n}.(\theta_1 \mid \{^{\langle U_1, U_2\rangle}/_{x}\})$ and $\psi = \nu\tilde{m}.(\theta_2 \mid \{^{\langle V_1, V_2\rangle}/_{x}\})$ two frames with U_1, U_2, V_1, V_2 some terms. We have:*

$$\phi \approx_s \psi \quad\Longleftrightarrow\quad \nu\tilde{n}.\big(\theta_1 \mid \big\{^{U_1}/_{x_1}, {}^{U_2}/_{x_2}\big\}\big) \approx_s \nu\tilde{m}.\big(\theta_2 \cup \big\{^{V_1}/_{x_1}, {}^{V_2}/_{x_2}\big\}\big).$$

Proof. Let $\phi' = \nu\tilde{n}.(\theta_1 \mid \{^{U_1}/_{x_1}, {}^{U_2}/_{x_2}\})$ and $\psi' = \nu\tilde{m}.(\theta_2 \cup \{^{V_1}/_{x_1}, {}^{V_2}/_{x_2}\})$.

Let M, N be terms s.t. $(M =_E N)\phi'$. We consider $\delta = [x_1 \mapsto \Pi_1(x), x_2 \mapsto \Pi_2(x)]$ and we have, for all term P: $(P\delta)\phi =_E P\phi'$ and $(P\delta)\psi =_E P\psi'$. Thus $M\phi' =_E N\phi'$ implies $(M\delta)\phi =_E (N\delta)\phi$. Since $\phi \approx_s \psi$ we deduce $(M\delta)\psi =_E (N\delta)\psi$, that is $M\psi' =_E N\psi'$. Since ϕ and ψ play a symmetric role, we deduce that $M\psi' =_E N\psi'$ implies $M\phi' =_E N\phi'$ as well. □

Adding a deducible term to the frames preserves static equivalence, provided the condition that the same recipe is used in the two frames.

Lemma 3. *Let $\tilde{n}, \tilde{m}$ be names, θ_1, θ_2 substitutions, and $\phi = \nu\tilde{n}.\theta_1$ and $\psi = \nu\tilde{m}.\theta_2$ frames. Let U be a free term, we have:*

$$\phi \approx_s \psi \quad\Longleftrightarrow\quad \nu\tilde{n}.\big(\theta_1 \mid \big\{^{U\theta_1}/_{x}\big\}\big) \approx_s \nu\tilde{m}.\big(\theta_2 \cup \big\{^{U\theta_2}/_{x}\big\}\big).$$

Proof. Let $\phi' = \nu\tilde{n}.(\theta_1 \mid \{^{U\theta_1}/_{x}\})$ and $\psi' = \nu\tilde{m}.(\theta_2 \cup \{^{U\theta_2}/_{x}\})$.

$\underline{\Longleftarrow}|$ Straightforward.

$\underline{\Longrightarrow}|$ Let M, N be terms such that $(M =_E N)\psi'$. We consider $\delta : x \mapsto U$ and we have, for all term P: $(P\delta)\phi =_E P\phi'$ and $(P\delta)\psi =_E P\psi'$. Thus $M\phi' =_E N\phi'$ implies $(M\delta)\phi =_E (N\delta)\phi$. Since $\phi \approx_s \psi$, we deduce $(M\delta)\psi =_E (N\delta)\psi$, that is $M\psi' =_E N\psi'$. □

The next lemmas hold for equational theories for which *destructors* can be identified.

Definition 5. Let E be an equational theory induced by an AC convergent rewriting system $\mathcal{R}$ and Σ a signature. We say that $f \in \Sigma$ is a destructor in E if for any rewrite rule $l \to r$ of $\mathcal{R}$

$$f \notin r \quad \text{and} \quad \big(f \notin l \text{ or } l = f(l_1, \ldots, l_n) \text{ and } \forall i \in [\![1, n]\!], f \notin l_i\big),$$

i.e. f can only appear in head in l and does not appear in r. Terms of the form $f(t_1, \ldots, t_n)$ with f destructor in E are called destructor terms.

The next lemma states that destructor terms cannot be introduced by rewriting rules.

Lemma 4. *Let E be an equational theory induced by an AC-convergent rewriting system $\mathcal{R}$, Σ a signature and $f \in \Sigma$ a destructor in E. Let $C[\_]$ be a context such that $f \notin C$ and let $T_1, \ldots, T_n$ be terms in normal form. Then there exists a context $C'[\_]$ such that $f \notin C'$ and $C[T_1, \ldots, T_n] \rightarrow C'[M_1, \ldots, M_p]$ where $M_1, \ldots, M_p$ are subterms of $T_1, \ldots, T_n$.*

Proof. Let us consider such context $C[\_]$ and such terms $T_1, \ldots, T_n$. For this proof, we also consider that we flatten AC symbols, i.e. a term $\oplus(x, \oplus(y, z))$ will be considered as $\oplus(x, y, z)$ for any AC symbol $\oplus$. Moreover, we can consider w.l.o.g. that each T_i does not start with an AC-symbol. If $T_i = \oplus(N_1, \ldots, N_q)$ for one $i \in [\![1, n]\!]$, then we can consider $\overline{C}[\_]$ a new context deduced from $C[\_]$ by adding a $\oplus$ symbol at each position T_i should be inserted in $C[\_]$. Then, we have:

$$C[T_1, \ldots, T_n] =_{AC} \overline{C}[T_1, \ldots, T_{i-1}, N_1, \ldots, N_q, T_{i+1}, \ldots, T_n].$$

Now, if $C[T_1, \ldots, T_n]$ is in normal form, the result is straightforward. If it is not, then we know that there exists a position s, a rule $l \rightarrow r$ from $\mathcal{R}$ and a substitution θ such that $C[T_1, \ldots, T_n]|_s =_{AC} l\theta$. Let x be a variable of l and p_x be a position such that $l|_{p_x} = x$. We know that p_x can't be the root, otherwise $l = x$ and $\mathcal{R}$ would contain a rule $x \rightarrow r$. Such a rule would allow infinite chains of reduction which is in contradiction with hypothesis on $\mathcal{R}$. Thus, we can move one step up and call p_s the position such that $p_x = p_s * j$ with $j \in \mathbb{N}^*$ and $l|_{p_s} = g \in \Sigma$. We consider two cases:

- g is not an AC-symbol. We consider two subcases:
  - p_x is a position of $C[\_]$. Then $x\theta =_{AC} C_x[T_1, \ldots, T_n]$ with $C_x[\_]$ a sub context of $C[\_]$ and we can replace the substitution of x in θ by $\{^{C_x[T_1,\ldots,T_n]}/_x\}$.
  - p_x is not a position of $C[\_]$, i.e. there exists p_i and p_2 two positions such that $p_x = p_i * p_2$ with $C[T_1, \ldots, T_n]|_{p_i} =_{AC} T_i$ and $T_i|_{p_2} =_{AC} x\theta =_{AC} W$ with $W \in \mathsf{St}(T_i)$. In that case, we can just replace the substitution of x in θ by $\{^W/_x\}$.
- $g = \oplus$, an AC-symbol. Then, we have $l|_{p_s} =_{AC} x \oplus l'$. We consider again two subcases:
  - Either p_s is a position of $C[\_]$. Then, $l|_{p_s}\theta =_{AC} (x \oplus l')\theta =_{AC} C_x[T_1, \ldots, T_n]$. Thus, $\mathsf{head}(C_x[T_1, \ldots, T_n)) = \oplus$ and we can expand it as follows:

    $$C_x[T_1, \ldots, T_n] =_{AC} \bigoplus_{i=1}^{m_1} C_x^i[T_1, \ldots, T_n] \bigoplus_{j=1}^{m_2} W_j$$

    with $C_x^i[\_]$ subcontexts of $C_x[\_]$ and W_j being one of $T_1, \ldots, T_n$. Then, each $x\theta$ is linked to a subset of this sum and we can replace the substitution of x in θ by $\{^{\bigoplus_{i=1}^{w_1} C_x^i[T_1,\ldots,T_n] \bigoplus_{j=1}^{w_2} W_j}/_x\}$ with $w_1 \leqslant m_1$ and $w_2 \leqslant m_2$.
  - Or $l|_{p_s}\theta =_{AC} W$ with $W \in \mathsf{St}(T_i)$ for one $i \in [\![1, n]\!]$. Then, $x\theta$ is subterm of W and we conclude.

Finally, for each case, we have that $l\theta =_{AC} l\theta'$ where θ' contain only substitutions of variables of l by sub contexts of $C[\_]$ and subterms of $T_1, \ldots, T_n$. Thus, we have:

$$C[T_1, \ldots, T_n]|_s =_{AC} l\theta \rightarrow r\theta' =_{AC} C_r[R_1, \ldots, R_{q_r}]$$

where $C_r[\_]$ contains r and all subcontexts $C_x[\_]$, $C_x^k[\_]$. Since $f \notin C$ and $C_x[\_]$, $C_x^k[\_]$ are subcontexts of $C[\_]$, we know that $f \notin C_x$ and $f \notin C_x^k$. Moreover, since f is a destructor in E, $f \notin r$. Thus, $f \notin C_r$. Finally, $R_1, \dots, R_{q_r}$ are subterms of $T_1, \dots, T_n$, then:

$$C[T_1, \dots, T_n] \to C[T_1, \dots, T_n]\big[C_r[R_1, \dots, R_{q_r}]\big]_p = C'[M_1, \dots, M_q]. \qquad \square$$

If a destructor term t is non deducible and does not appear as subterm of a frame then t cannot appear as subterm of any deducible term.

Lemma 5. *Let E be an equational theory induced by an AC convergent rewriting system $\mathcal{R}$, Σ a signature and $f \in \Sigma$ is a destructor in E. Let φ be a frame and $U = f(U_1, \dots, U_n)$ a term in normal form such that $U \notin \mathsf{St}(\varphi)$ and $\varphi \nvdash U$. Then:*

for all term T in normal form such that $\varphi \vdash T$, $\quad U \notin \mathsf{St}(T)$.

Proof. Let us prove this by induction on the number of steps used to deduce T.

Base case: $T \in \varphi$. Then, since $U \notin \mathsf{St}(\varphi)$, we have that $U \notin \mathsf{St}(T)$.

Induction case: Now suppose that for any T in normal form such that $\varphi \vdash T$ in n steps, then $U \notin \mathsf{St}(T)$. Let T in normal form such that $\varphi \vdash T$ in $n+1$ steps. We have $\varphi \vdash T =_{AC} g(T_1, \dots, T_n)\downarrow$ where $g \in \Sigma$ and $\varphi \vdash T_i$ in n steps with T_i in normal form for $i \in [\![1, n]\!]$.

- If $g(T_1, \dots, T_n)$ is already in normal form, then we have two subcases:
  - ∗ If $g \neq f$, then according to the induction hypothesis and since $U \notin \mathsf{St}(T_i)$, we have $U \notin \mathsf{St}(T)$.
  - ∗ If $g = f$, then, if $U \in \mathsf{St}(T)$ and since $U \notin \mathsf{St}(T_i)$ according to the induction hypothesis, we must have $T = U$, which yields to a contradiction since $\varphi \vdash T$ and $\varphi \nvdash U$.
- If $g(T_1, \dots, T_n)$ is not in normal form. If $g = f$ then, we have that $f(T_1, \dots, T_n) \to r[\widetilde{T_i}]$ for some rule $l \to r$. Since f is a destructor in E, we know that $f \notin r$. Thus, according to Lemma 4, we know that $r[\widetilde{T_i}] \to^* C[\widetilde{T_i}]$ such that $f \notin C$. In that case, if $U \in \mathsf{St}(C[\widetilde{T_i}])$, then, we must have $U \in \mathsf{St}(\widetilde{T_i})$ which is a contradiction. If $g \neq f$, then, using Lemma 4 with $C[T_1, \dots, T_n] =_{AC} g(T_1, \dots, T_n)$, we have that $\exists C'$ such that $f \notin C'$ and $g(T_1, \dots, T_n) \to^* C'[\widetilde{T_i}]$ with $\widetilde{T_i}$ subterms of $T_1, \dots, T_n$. Then $T =_{AC} C'[\widetilde{T_i}]$ but we know that $U \notin \mathsf{St}(\widetilde{T_i})$ according to induction hypothesis, thus $U \notin \mathsf{St}(T)$.

This concludes the induction. □

A context in normal form applied to a destructor term remains in normal form.

Lemma 6. *Let E be an equational theory induced by an AC convergent rewriting system $\mathcal{R}$, Σ a signature and $f \in \Sigma$ is a destructor in E. Let $U = f(U_1, \dots, U_n)$ a term in normal form and $P[X]_p$ another term in normal form. Then $P[U]_p$ is also in normal form.*

Proof. Assume that $P[U]$ is not in normal form. Since $P[X]$ and U are in normal form, it implies that $\exists p$ a position of $P[\_]$ such that $P[U]|_p =_{AC} l\theta$ with some rule $l \to r$ and some substitution θ. But f is a destructor in E, thus, f must appear at the head of l, which is a contradiction with the position p. □

As a consequence of the previous lemmas, we can state that adding non deducible destructor terms to the frames preserve static equivalence.

Lemma 7. *Let E be an equational theory, Σ a signature and $f, g \in \Sigma$ destructors in E. Let $\varphi_1 = \nu\tilde{\omega}_1.\theta_1$ and $\varphi_2 = \nu\tilde{\omega}_2.\theta_2$ be frames. Let $U_1 = f(U_1^1, \ldots, U_p^1)$ and $U_2 = g(U_1^2, \ldots, U_q^2)$ be terms in normal form such that, $\forall i \in \{1, 2\}$, $\varphi_i \nvdash U_i$ and $U_i \notin \mathsf{St}(\varphi_i)$. Then:*

$$\varphi_1 \approx_s \varphi_2 \quad\Longleftrightarrow\quad \nu\tilde{\omega}_1.\big(\theta_1 \mid \big\{^{U_1}/_x\big\}\big) \approx_s \nu\tilde{\omega}_2.\big(\theta_2 \mid \big\{^{U_2}/_x\big\}\big).$$

Proof. Let $\varphi_i' = \nu\tilde{\omega}_i.(\theta_i \mid \{^{U_i}/_x\}) = \nu\tilde{\omega}_i.\theta_i'$ for $i \in \{1, 2\}$.

$\underline{\Longleftarrow|}$ Straightforward.

$\underline{\Longrightarrow|}$ Let M and N be terms s.t. $\mathsf{fv}(M, N) \subseteq \mathsf{dom}(\varphi_1')$ and $\mathsf{bn}(M, N) \cap \mathsf{bn}(\varphi_1') = \emptyset$ with $(M =_E N)\varphi_1'$. Then:

$$\begin{array}{rcl} M\varphi_1' & =_E & N\varphi_1' \\ \big(M\varphi_1'\big)\!\downarrow & =_{AC} & \big(N\varphi_1'\big)\!\downarrow \\ (M\varphi_1\delta_1)\!\downarrow & =_{AC} & (N\varphi_1\delta)\!\downarrow \end{array}$$

with $\delta_1 : x \mapsto U_1$, according to φ_1' definition. Since U_1 is in normal form, we apply Lemma 6:

$$\begin{array}{rcl} (M\varphi_1)\!\downarrow \delta_1 & =_{AC} & (N\varphi_1)\!\downarrow \delta_1 \\ \big((M\varphi_1)\!\downarrow \delta_1\big)\delta_1^{-1} & =_{AC} & \big((N\varphi_1)\!\downarrow \delta_1\big)\delta_1^{-1} \end{array}$$

with $\delta_1^{-1} : U_1 \mapsto x$. We know that $U_1 = f(U_1^1, \ldots, U_p^1)$ with f destructor in E and U_1 in normal form and not subterm of φ_1. Thus, using Lemma 5, for any T in normal form s.t. $\varphi_1 \vdash T$, then $U_1 \notin \mathsf{St}(T)$. And, since $\varphi_1 \vdash (M\varphi_1)\!\downarrow$ (resp. $(N\varphi_1)\!\downarrow$) we have that $(M\varphi_1)\!\downarrow \delta_1^{-1} = (M\varphi_1)\!\downarrow$ (resp. $(N\varphi_1)\!\downarrow \delta_1^{-1} = (N\varphi_1)\!\downarrow$). Thus:

$$\begin{array}{rcl} (M\varphi_1)\!\downarrow \delta_1\delta_1^{-1} & =_{AC} & (N\varphi_1)\!\downarrow \delta_1\delta_1^{-1} \\ (M\varphi_1)\!\downarrow & =_{AC} & (N\varphi_1)\!\downarrow \end{array}$$

Since $\varphi_1 \approx_s \varphi_2$, we have (with $\delta_2 : x \mapsto U_2$):

$$\begin{array}{rcl} (M\varphi_2)\!\downarrow & =_{AC} & (N\varphi_2)\!\downarrow \\ (M\varphi_2)\!\downarrow \delta_2 & =_{AC} & (N\varphi_2)\!\downarrow \delta_2 \end{array}$$

Using Lemma 6, we deduce $(M\varphi_2)\!\downarrow \delta_2 =_{AC} (N\varphi_2)\!\downarrow$, implying that $(M =_E N)\varphi_1' \rightarrow (M =_E N)\varphi_2'$. Repeating the same reasoning, we can also prove that $(M =_E N)\varphi_2' \rightarrow (M =_E N)\varphi_1'$, and we conclude. □

5.2. *A more specific lemma*

We show that adding the signature of a deducible term preserves static equivalence, when the same recipe is used in both frames. We believe that this lemma holds for other primitives provided the equations are similar to those for the “sign symbol like the case of zero-knowledge proofs.

Lemma 8. *Let $\varphi_1 = \nu\omega_1.\theta_1$ and $\varphi_2 = \nu\omega_2.\theta_2$ be two frames, x a fresh variable and a, a name such that $a \in \omega_1 \cap \omega_2$, $\{^{\mathsf{vk}(a)}/_{\mathsf{idp_a}}\} \in \theta_1 \cap \theta_2$ and $\varphi_1, \varphi_2 \nvdash a$. Let $U = \mathsf{sign}(U', a)$ in normal form with U' a free term and such that $(U\varphi_i)\!\downarrow \notin \mathsf{St}(\varphi_i)$. Then:*

$$\varphi_1 \approx_s \varphi_2 \quad\Longleftrightarrow\quad \nu\omega_1.\big(\theta_1 \mid \big\{^{(U\varphi_1)\downarrow}/_x\big\}\big) \approx_s \nu\omega_2.\big(\theta_2 \mid \big\{^{(U\varphi_2)\downarrow}/_x\big\}\big).$$

Before proving this lemma, we introduce the notion of down-to-top strategy, where a rewrite rule is applied first as deep as possible. Note that since we consider convergent rewrite systems, we may chose any rewrite strategy.

Definition 6. Let us consider $M_0 \xrightarrow{l_1 \to r_1} M_1 \longrightarrow \cdots \xrightarrow{l_i \to r_i} M_i \cdots \longrightarrow M_n = M_0{\downarrow}$ a reduction strategy. This strategy is called *down-to-top*, noted $\longrightarrow_{dt}$, if, for all $i \in [\![1, n]\!]$, $M_{i-1}|_{p_i} =_{AC} l_i\theta_i \to r_i\theta_i$ and for all position $q < p_i$, $M_{i-1}|_q$ is in normal form.

Proof. Let $\overline{\varphi}_i = \nu\omega_1.\overline{\theta}_i$ with $\overline{\theta}_i = \theta_i \mid \{^{(U\varphi_i)\downarrow}/_x\}$ for $i \in \{1, 2\}$.
$\underline{\Longleftarrow|}$ Straightforward.
$\underline{\Longrightarrow|}$ We first assume the two following claims that we prove next.

Claim 1. *Let M be a term s.t. $\mathsf{fn}(M) \cap \omega_i = \emptyset$. Then, for any term T s.t. $M\varphi_i \longrightarrow^*_{dt} T$, we have $(U\varphi_i){\downarrow} \notin \mathsf{St}(T)$.*

Claim 2. *Let M be a term s.t. $(U\varphi_i){\downarrow} \notin \mathsf{St}(M)$ and for all term T s.t. $M \longrightarrow^*_{dt} T$, then $(U\varphi_i){\downarrow} \notin \mathsf{St}(T)$. Let $\sigma = \{^{(U\varphi_i)\downarrow}/_x\}$ a substitution. If $M\sigma \longrightarrow^*_{dt} T$ with no rule (8) involved in the reduction path, then there exists some term T' such that $M \longrightarrow^*_{dt} T'$ and $T'\sigma =_{AC} T$.*

We now suppose that $\varphi_1 \approx_s \varphi_2$ and we consider M and N two terms s.t. $\mathsf{fv}(M, N) \subseteq \mathsf{dom}(\overline{\varphi}_1)$ and $\mathsf{fn}(M, N) \cap \omega_1 = \emptyset$. (Since $\varphi_1 \approx_s \varphi_2$, we have $\omega_1 = \omega_2$ and $\mathsf{dom}(\overline{\varphi}_1) = \mathsf{dom}(\overline{\varphi}_2)$.) We are going to prove that $(M =_E N)\overline{\varphi}_1$ iff $(M =_E N)\overline{\varphi}_2$ by induction on the number of positions p in M and N s.t. $M|_p = \mathsf{checksign}(M_1, M_2, M_3)$ and $(M|_p\overline{\varphi}_i){\downarrow} =_E \mathsf{ok}$ for $i = 1$ or $i = 2$. We also assume down-to-top reduction strategies only (noted $\longrightarrow_{dt}$).

Base case: There is no such position in M and N. Then, we can apply Claim 2 on $M\overline{\varphi}_i$ and $N\overline{\varphi}_i$. Indeed, let $\sigma_i = \{^{(U\varphi_i)\downarrow}/_x\}$. For $P \in \{M, N\}$, then $P\overline{\varphi}_i = (P\varphi_i)\sigma_i \longrightarrow^*_{dt} (P\overline{\varphi}_i){\downarrow}$. If there exists a position p such that $(P\varphi_i)|_p =_{AC} (U\varphi_i){\downarrow}$ then, according to the hypothesis on φ_i, p must be a position of P (and not at a leaf). This implies that $P|_{p*2} = a$ which would be a contradiction with the fact that a is restricted. Thus, we have $(U\varphi_i){\downarrow} \notin \mathsf{St}(P\varphi_i)$. We also have $P\varphi_i \longrightarrow^*_{dt} (P\varphi_i){\downarrow}$ and s.t. $\mathsf{fn}(P) \cap \omega_i = \emptyset$. Using Claim 1, we deduce that $(U\varphi_i){\downarrow}$ does not occur in any of the terms in the reduction $P\varphi_i \longrightarrow^*_{dt} (P\varphi_i){\downarrow}$. Applying Claim 2 leads to $(P\overline{\varphi}_i){\downarrow} =_{AC} (P\varphi_i){\downarrow}\,\sigma_i$. Then:

$$
\begin{array}{rcll@{\qquad}rcll}
(M\overline{\varphi}_i){\downarrow} & =_{AC} & (N\overline{\varphi}_i){\downarrow} & & (M\varphi_2){\downarrow} & =_{AC} & (N\varphi_2){\downarrow} & (\varphi_1 \approx_s \varphi_2) \\
(M\varphi_1){\downarrow}\,\sigma_1 & =_{AC} & (N\varphi_1){\downarrow}\,\sigma_1 & (\text{Claim } 2) & (M\varphi_2){\downarrow}\,\sigma_2 & =_{AC} & (N\varphi_2){\downarrow}\,\sigma_2 & \\
(M\varphi_1){\downarrow} & =_{AC} & (N\varphi_1){\downarrow} & (\text{Claim } 1) & (M\overline{\varphi}_2){\downarrow} & =_{AC} & (N\overline{\varphi}_2){\downarrow}\,. & (\text{Claim } 2)
\end{array}
$$

Induction step: We suppose that there exists at least one position p in M or N s.t. $M|_p = \mathsf{checksign}(M_1, M_2, M_3)$ (or $N|_p$) and $(M|_p\overline{\varphi}_i){\downarrow} =_E \mathsf{ok}$ (resp. $N|_p$) for $i = 1$ or $i = 2$. Let us consider w.l.o.g. that this position is in M and that it reduces when $\overline{\varphi}_1$ is applied. (We note that if $(M|_p\overline{\varphi}_2){\downarrow} =_E \mathsf{ok}$ too then we have $(M[\mathsf{ok}]_p =_E M)\overline{\varphi}_i$ for $i \in \{1, 2\}$ and we can conclude using the induction hypothesis on $M[\mathsf{ok}]_p$ and N.) We consider the deepest position satisfying this in M. Thus, we know that M_1, M_2 and M_3 do not contain such a position.

Since $(M|_p\overline{\varphi}_1){\downarrow} =_E \mathsf{ok}$, then, according to E, we have that $(M_3\overline{\varphi}_1){\downarrow} =_{AC} \mathsf{sign}((M_1\overline{\varphi}_1){\downarrow}, Q)$ and $(M_2\overline{\varphi}_1){\downarrow} =_{AC} \mathsf{vk}(Q)$ for some term Q. According to the hypothesis on position p, we can apply the same reasoning as used in the base case on M_1, M_2 and M_3 separately and therefore we can apply

Claim 2 on them, which leads to $(M_i\overline{\varphi}_1)\!\downarrow =_{AC} (M_i\varphi_1)\!\downarrow \sigma_1$ for $i \in \{1, 2, 3\}$. We have then the following equalities:

- $(M_1\varphi_1)\!\downarrow =_{AC} T_1$ and $T_1\sigma_1 =_{AC} (M_1\overline{\varphi}_1)\!\downarrow$.
- $(M_2\varphi_1)\!\downarrow =_{AC} T_2$ and $T_2\sigma_1 =_{AC} \mathsf{vk}(Q)$. According to definition of σ_1, which does not contain a term of this form, we must have $T_2 =_{AC} \mathsf{vk}(T_2')$ and $T_2'\sigma_2 =_{AC} Q$.
- $(M_3\varphi_1)\!\downarrow =_{AC} T_3$ and $T_3\sigma_1 =_{AC} \mathsf{sign}((M_1\overline{\varphi}_1)\!\downarrow, Q)$. Using the two previous equalities, we have $T_3\sigma_1 =_{AC} \mathsf{sign}(T_1\sigma_1, T_2'\sigma_1)$. Then, according to the definition of σ_1, we consider two subcases:

  - $*$ $T_3 =_{AC} \mathsf{sign}(T_1, T_2',)$ and then, we have:

$$\begin{aligned}(M|_p\varphi_1)\!\downarrow &=_E \mathsf{checksign}\big((M_1\varphi_1)\!\downarrow, (M_2\varphi_1)\!\downarrow, (M_3\varphi_1)\!\downarrow\big)\!\downarrow\\ &=_E \mathsf{checksign}\big(T_1, \mathsf{vk}\big(T_2'\big), \mathsf{sign}\big(T_1, T_2'\big)\big)\!\downarrow\\ &=_E \mathsf{ok}.\end{aligned}$$

    Thus, $(M|_p =_E \mathsf{ok})\varphi_1$ and using $\varphi_1 \approx_s \varphi_2$, we have $(M|_p =_E \mathsf{ok})\varphi_2$ which also implies by extension $(M|_p =_E \mathsf{ok})\overline{\varphi}_2$ and we conclude.

  - $*$ $T_3 = x$. Then, we have $T_2\sigma_1 =_{AC} \mathsf{vk}(a)$, i.e. $T_2 =_{AC} \mathsf{vk}(a)$, which is $(M_2\varphi_1)\!\downarrow =_{AC} \mathsf{vk}(a)$. Thus, we have $(M_2 =_E \mathsf{idp_a})\varphi_1$ and, using $\varphi_1 \approx_s \varphi_2$ that leads us to $(M_2 =_E \mathsf{idp_a})\overline{\varphi}_2$. Moreover, $(M_1\overline{\varphi}_1)\!\downarrow =_{AC} T_1\sigma_1 =_{AC} (U'\varphi_1)\!\downarrow$, i.e. $T_1 =_{AC} (U'\varphi_1)\!\downarrow$ according to the definition of σ_1, which implies $(M_1\varphi_1)\!\downarrow =_{AC} (U'\varphi_1)\!\downarrow$. Using $\varphi_1 \approx_s \varphi_2$ again, we have $(M_1 =_E U')\overline{\varphi}_2$. Then:

$$\begin{aligned}(M|_p\overline{\varphi}_2)\!\downarrow &=_E \mathsf{checksign}\big((M_1\overline{\varphi}_2)\!\downarrow, (M_2\overline{\varphi}_2)\!\downarrow, (x\overline{\varphi}_2)\!\downarrow\big)\!\downarrow\\ &=_E \mathsf{checksign}\big(\big(U'\varphi_2\big)\!\downarrow, (\mathsf{idp_a}\varphi_2)\!\downarrow, (U\varphi_2)\!\downarrow\big)\!\downarrow\\ &=_E \mathsf{ok}.\end{aligned}$$

    And we conclude.

We now prove that the two claims hold.

Proof of Claim 1 We prove this Claim by induction on the size of M. We suppose w.l.o.g. that M is in normal form.

Base case: $M = y$ is a variable (or a name). Since φ_i is in normal form, we have that for all T s.t. $M\varphi_i \longrightarrow_{dt}^{*} T$ then $M\varphi_i =_{AC} T$. If M is a name, result is straightforward. If M is a variable, then $(U\varphi_i)\!\downarrow \in \mathsf{St}(T)$ implies that $(U\varphi_i)\!\downarrow \in \mathsf{St}(\varphi_i)$ which is a contradiction.

Induction Step: We assume that for any term M of size m s.t. $\mathsf{fn}(M) \cap \omega_i = \emptyset$ we have, for all term T s.t. $M\varphi_i \longrightarrow_{dt}^{*} T$, $(U\varphi_i)\!\downarrow \notin \mathsf{St}(T)$. We now consider a term M of size $(m + 1)$, i.e. $M = \mathsf{f}(M_1, \dots, M_n)$ with M_i of size at most equal to m. Moreover, following a down-to-top reduction strategy, we have $M\varphi_i = \mathsf{f}(M_1\varphi_i, \dots, M_n\varphi_i) \longrightarrow_{dt}^{*} \mathsf{f}(T_1, \dots, T_n)$ with $T_j = (M_j\varphi_i)\!\downarrow$. According to the induction hypothesis, we now that for all term T s.t. $M_j\varphi_i \longrightarrow_{dt}^{*} T$, then $(U\varphi_i)\!\downarrow \notin \mathsf{St}(T)$. Let us consider now the down-to-top reduction strategy $M\varphi_i \longrightarrow_{dt}^{*} T = \mathsf{f}(T_1', \dots, T_n') \longrightarrow_{dt}^{*} \mathsf{f}(T_1, \dots, T_n)$. If $(U\varphi_i)\!\downarrow \in \mathsf{St}(T)$, then we have two possibilities. Either $(U\varphi_i)\!\downarrow \in \mathsf{St}(T_k')$ for some $k \in [\![1, n]\!]$ and we have a contradiction with the induction hypothesis; or $\mathsf{f}(T_1', \dots, T_n') =_{AC} (U\varphi_i)\!\downarrow$ and $\mathsf{f} = \mathsf{sign}$, $n = 2$ and $T_2' = \mathsf{a}$ which implies that a is deducible by φ_i, contradiction. Thus, $(U\varphi_i)\!\downarrow$ does not occur in any of the terms in the reduction $(M\varphi_i)\!\downarrow \longrightarrow_{dt}^{*} \mathsf{f}(T_1, \dots, T_n)$. We now consider the reduction of $\mathsf{f}(T_1, \dots, T_n)$:

- It is in normal form, then, we have two subcases:

  - $*$ $\mathsf{f} \neq \mathsf{sign}$. Then, since $(U\varphi_i)\!\downarrow \notin \mathsf{St}(T_j)$ for $j \in [\![1,n]\!]$ by the induction hypothesis, we conclude.
  - $*$ $\mathsf{f} = \mathsf{sign}$. Using the induction hypothesis, if $(U\varphi_i)\downarrow \in \mathsf{St}(\mathsf{sign}(T_1,T_2))$, we must have $\mathsf{sign}(T_1,T_2) =_{AC} (U\varphi_i)\!\downarrow$ and a would be deducible by φ_i which is a contradiction.

- $\mathsf{f}(T_1,\dots,T_n)$ is not in normal form. Then, we have $\mathsf{f} \neq \mathsf{sign}$ since there are no rule in E s.t. sign is in head. We have $\mathsf{f}(T_1,\dots,T_n) \longrightarrow_{dt} T$ and we consider two subcases:
  - $*$ $T \in \mathsf{St}(T_1,\dots,T_n)$, then T is in normal form and, using the induction hypothesis, we know that $(U\varphi_i)\!\downarrow \notin \mathsf{St}(T)$ and we conclude.
  - $*$ $T \notin \mathsf{St}(T_1,\dots,T_n)$, then, according to E, we have used rule (4), (5), (6) or (8), (9), (10). The three last rules lead to $T = \mathsf{ok}$ which concludes straightforwardly and the three first rules lead to T in normal form with the property that if $(U\varphi_i)\!\downarrow \in \mathsf{St}(T)$, then $(U\varphi_i)\!\downarrow \in \cup\mathsf{St}(T_k)$, which would be a contradiction.

Proof of Claim 2 Let us prove this by induction on the number of reduction steps in $M\sigma \longrightarrow_{dt}^{*} T$.

Base case: If $M\sigma =_{AC} T$, the result is straightforward. If $M\sigma \longrightarrow_{dt} T$ in one step, then, there is a position p, a substitution θ and a rule $l \longrightarrow r$ (different from (8)) s.t. $(M\sigma)|_p =_{AC} l\theta \longrightarrow_{dt} r\theta$. Since σ is in normal form, p must be a position of M. In particular, if we consider $M' = M[z]_p$, we have:

$$M\sigma =_{AC} \big(M[z]_p\sigma\big)[l\theta]_p =_{AC} \big(M'\sigma\big)[l\theta]_p \quad \text{and} \quad M\sigma \longrightarrow_{dt} \big(M'\sigma\big)[r\theta]_p.$$

We note $\delta : (U\varphi_i)\!\downarrow \mapsto x$. Then, we have $(M\sigma)\delta =_{AC} ((M'\sigma)[l\theta]_p)\delta$. Hypothesis on M states that there is no position q such that $M|_q =_{AC} (U\varphi_i)\!\downarrow$. Therefore, there is no position q such that $M'|_q =_{AC} (U\varphi_i)\!\downarrow$ and the following equalities hold: $(M\sigma)\delta =_{AC} M(\sigma\delta) =_{AC} M$ and $(M'\sigma)\delta =_{AC} M'(\sigma\delta) =_{AC} M'$. Moreover, we know that $(l\theta) \notin \mathsf{St}((U\varphi_i)\!\downarrow)$ otherwise $(U\varphi_i)\!\downarrow$ would not be in normal form. Thus, $M =_{AC} M'[(l\theta)\delta]_p$. Since $l \longrightarrow r$ is different from rule (8), we have $(l\theta)\delta = l(\theta\delta) = l\theta' \longrightarrow r\theta'$, for $\theta' = \theta\delta$, and this implies $M \longrightarrow_{dt} M'[r\theta']_p$ following a down-to-top reduction strategy since if there was a lower reduction in M it would also exists in $M\sigma$. Now, we can conclude since:

$$\begin{aligned}\big(M'[r\theta']_p\big)\sigma &=_{AC} \big(M'\sigma\big)\big[(r\theta')\sigma\big]_p\\ &=_{AC} \big(M'\sigma\big)\big[r\big(\theta'\sigma\big)\big]_p\\ &=_{AC} \big(M'\sigma\big)[r\theta]_p.\end{aligned}$$

Induction Step: We suppose that, for any term M s.t. for all term T s.t. $M \longrightarrow_{dt}^{*} T$, then $(U\varphi_i)\!\downarrow \notin \mathsf{St}(T)$, we have: if $M\sigma \longrightarrow_{dt}^{m} T$ in m steps involving no (8) rule, then $M \longrightarrow_{dt}^{m} T'$ with $T'\sigma =_{AC} T$. We consider now M s.t. $M\sigma \longrightarrow_{dt}^{m+1} T$ in $(m+1)$ steps with no rules (8) in the reduction path. Then $M\sigma \longrightarrow_{dt}^{m} M_1 \longrightarrow_{dt} T$. Using the induction hypothesis, we have that $M \longrightarrow_{dt}^{m} M_1'$ with $M_1'\sigma =_{AC} M_1$, thus $M_1'\sigma \longrightarrow_{dt} T$. But M' is verifying hypothesis of Claim 2, otherwise we would have a contradiction with the fact that M is satisfying them. Then, using Base Case, we have $M_1' \longrightarrow_{dt} T'$ with $T'\sigma =_{AC} T$. Thus, $M \longrightarrow_{dt}^{m} M_1' \longrightarrow_{dt} T'$ and $T'\sigma =_{AC} T$, which allows us to conclude. □

6. Static equivalence of the final frame

This section is devoted to the proof of Proposition 1, that states static equivalence of the final frame obtained in the relation $\mathcal{R}$. This proof relies on several lemmas. Some of them have been presented in the previous section, other are stated here and proved in appendix.

6.1. Definitions and useful lemmas

We first start by introducing some notations that will be useful for the following lemmas. Some of these definitions may seem artificial but mainly come from the establishment of the relation $\mathcal{R}$, which is described in the Appendix Section F.

Definition 7. We introduce (or remind) the following definitions:

$$\begin{aligned}
\tilde{\omega} &= \mathsf{a}_1, \mathsf{a}_3, \mathsf{id}_1, \mathsf{id}_2, \mathsf{t}_1, \mathsf{t}_2, \mathsf{id}_\mathsf{R},\\
\theta_{\mathsf{init}} &= \left\{{}^{\mathsf{vk}(\mathsf{id}_\mathsf{k})}/_{\mathsf{idp}_\mathsf{k}}, {}^{\mathsf{s}(\mathsf{id}_\mathsf{k})}/_{\mathsf{s}_\mathsf{k}} \mid \mathsf{k} = 1, \ldots, n\right\} \mid \left\{{}^{\mathsf{vk}(\mathsf{id}_\mathsf{R})}/_{\mathsf{idp}_\mathsf{R}}\right\} \mid \left\{{}^{\mathsf{pk}(\mathsf{a}_\mathsf{k})}/_{\mathsf{g}_\mathsf{k}} \mid \mathsf{k} = 1 \ldots, 3\right\},\\
\theta_0 &= \theta_{\mathsf{init}} \mid \left\{{}^{\mathsf{penc}(\mathsf{v}_\mathsf{k},\mathsf{r}_\mathsf{k},\mathsf{g}_1)}/_{\mathsf{e}_\mathsf{k}}, {}^{\mathsf{pfk}_1(\mathsf{id}_\mathsf{k},\mathsf{t}_\mathsf{k},\mathsf{v}_\mathsf{k},\mathsf{e}_\mathsf{k})}/_{\mathsf{p}_\mathsf{k}}, {}^{\mathsf{sign}(\langle \mathsf{e}_\mathsf{k},\mathsf{p}_\mathsf{k}\rangle,\mathsf{id}_\mathsf{k})}/_{\mathsf{si}_\mathsf{k}} \mid \mathsf{k} = 1, 2\right\},\\
\theta_k &= \theta_{k-1} \mid \left\{{}^{\mathsf{sign}(\mathsf{hash}(\Pi_1(M_\mathsf{k})),\mathsf{id}_\mathsf{R})}/_{sr_k}, {}^{\mathsf{d}(\mathsf{p}(\mathsf{id}_\mathsf{k}),\mathsf{dec}(\Pi_2(M_\mathsf{k}),\mathsf{a}_3))}/_{rec_k}\right\},\\
\theta_\delta &= \theta_n \mid \left\{{}^{\mathsf{dec}(U_{\delta(k)},\mathsf{a}_1)}/_{res_k} \mid \mathsf{k} = 1, \ldots, n\right\},\\
\sigma_\mathsf{L} &= \left\{{}^{\mathsf{a}}/_{\mathsf{v}_1}, {}^{\mathsf{b}}/_{\mathsf{v}_2}\right\}, \qquad \sigma_\mathsf{R} = \left\{{}^{\mathsf{b}}/_{\mathsf{v}_1}, {}^{\mathsf{a}}/_{\mathsf{v}_2}\right\},\\
\hat{\sigma} &= \left\{{}^{M_\mathsf{k}\alpha}/_{x_k}, {}^{U_\mathsf{k}\alpha}/_{d_k}, {}^{W_\mathsf{k}\alpha}/_{hb_k} \mid \mathsf{k} = 1, \ldots, n\right\} \mid \left\{{}^{N_\mathsf{k}\alpha}/_{x_b^k} \mid \mathsf{k} = 1, \ldots, 2\right\},\\
\hat{\sigma}_i^j &= \left\{{}^{M_\mathsf{k}\alpha}/_{x_k} \mid \mathsf{k} = 1, \ldots, i\right\} \mid \left\{{}^{N_\mathsf{k}\alpha}/_{x_b^k} \mid \mathsf{k} = 1, \ldots, \mathsf{min(i, 2)}\right\} \mid \left\{{}^{U_\mathsf{k}\alpha}/_{d_k}, {}^{W_\mathsf{k}\alpha}/_{hb_k} \mid \mathsf{k} = 1, \ldots, j\right\}.
\end{aligned}$$

where α is a substitution representing the intermediate outputs visible by an adversary controlling the corrupted Ballot Box (full definition given in Section A), and M_k, N_k, U_k, W_k are free terms satisfying some conditions defined in Section F. They represent the inputs submitted by the adversary during the different steps of the protocol.

The first lemma ensures that several secret datas (like encryption/decryption keys, voters' secret IDs, etc.) remain secret during the execution of the protocol.

Lemma 9. *For* $\mathsf{i} \in \{1, 2\}$, $\mathsf{j} \in \{1, 2, 3\}$ *and any free term* U*, if* M *is of one of the following forms:* $\mathsf{a_j} + U$, $\mathsf{t_i} * U$, $\mathsf{id_i}$ *or* $\mathsf{p}(\mathsf{id_i})$*, then* $\nu\tilde{\omega}.\theta_\delta \nvdash M$*, i.e.* M *is not deducible from the frame* $\nu\tilde{\omega}.\theta_\delta$.

We then prove that the different outputs of the (honest) Receipt Generator, that is, a signature and a receipt for each submitted ballot, do not bring any relevant information to the adversary that could help him to make a difference between two different executions of the protocol. This property is expressed by the fact that we can add these terms to the frame without breaking the static equivalence.

Lemma 10. *For* $i \in [\![0, n]\!]$*, we have:* $\nu\tilde{\omega}.\theta_i\hat{\sigma}_i^0\sigma_\mathsf{L} \approx_s \nu\tilde{\omega}.\theta_i\hat{\sigma}_i^0\sigma_\mathsf{R}$.

Finally, we show that we can control the form of what is sent to the Decryption Device, assuming it is approved by the Auditor.

Lemma 11. *Let us consider* $M_1, \ldots, M_n$ *free terms s.t.* $\forall i \in [\![1, n]\!]$, $\phi_\mathsf{R}(\mathsf{idp_{i+1}}, M_{i+1})\theta_i\hat{\sigma}_i^0\sigma_\mathsf{L} = \mathsf{ok}$ *with* $\mathsf{fv}(M_{i+1}) \subseteq \mathsf{dom}(\theta_i) \cup \mathsf{dom}(\hat{\sigma}_i^0) \setminus \{x_b^i\}$*. We suppose that* $\forall i \in [\![1, n]\!]$, $\nu\tilde{\omega}.\theta_i\hat{\sigma}_i^0\sigma_\mathsf{L} \approx_s \nu\tilde{\omega}.\theta_i\hat{\sigma}_i^0\sigma_\mathsf{R}$*. We also consider* $U_1, \ldots, U_n$ *and* $W_1, \ldots, W_n$ *be free terms such that:*

- $\mathsf{fv}(U_{k+1}) \subseteq \mathsf{dom}(\theta_n) \cup \{\mathsf{dom}(\hat{\sigma}_n^0)\} \cup \{d_1, \dots, d_k\}$ *and* $\mathsf{fv}(W_{k+1}) \subseteq \mathsf{dom}(\theta_n) \cup \{\mathsf{dom}(\hat{\sigma}_n^k)\}$,
- $\phi_{\mathsf{A}}(\mathsf{hash}(\langle U_1, \dots, U_n\rangle), hb_r^1, \dots, hb_r^n, W_1, \dots, W_n)\theta_n\hat{\sigma}\sigma_{\mathsf{L}} = \mathsf{ok}$, *with the corresponding notation* $hb_r^i = \langle \mathsf{idp_i}, \mathsf{hash}(\Pi_1(M_i))\rangle$.

Then, we have, for $\sigma \in \{\sigma_{\mathsf{L}}, \sigma_{\mathsf{R}}\}$, $\forall 1 \leqslant i \leqslant n$*:*

$$U_i\theta_n\hat{\sigma}\sigma =_E \mathsf{e_i}\theta_n\hat{\sigma}\sigma, \quad \textit{for } i = 1, 2.$$

Moreover, there exist free terms U_1', U_2' *and free term* U_3' *or* $U_3' = \mathsf{pk}(\mathsf{a_k} + U)$ *with free term* U*:*

$$U_i\theta_n\hat{\sigma}\sigma =_E \mathsf{penc}\big(U_1', U_2', U_3'\big)\theta_n\hat{\sigma}\sigma, \quad \textit{for } i \in [\![3, n]\!].$$

6.2. *Proving Proposition 1*

With all these lemmas, we can now prove the final static equivalence, which is stated in Proposition 1.

Proof. We introduce the notation $\hat{\theta}_i^\delta = \theta_n \mid \{^{dec_{\delta(k)}}/_{res_k} \mid k \in [\![1, i]\!]\}$ with $dec_{\delta(i+1)} = \mathsf{dec}(d_{\delta(i+1)}, \mathsf{a_1})$. We show by induction that:

$$\nu\tilde{\omega}.\hat{\theta}_i^\delta\hat{\sigma}\sigma_{\mathsf{L}} \approx_s \nu\tilde{\omega}.\hat{\theta}_i^{t\delta}\hat{\sigma}\sigma_{\mathsf{R}}.$$

Base case. The base case is ensured by Lemma 10 which guarantees that $\nu\tilde{\omega}.\theta_n\hat{\sigma}_n^0\sigma_{\mathsf{L}} \approx_s \nu\tilde{\omega}.\theta_n\hat{\sigma}_n^0\sigma_{\mathsf{R}}$ thus, by rewriting, we have $\nu\tilde{\omega}.\hat{\theta}_0^\delta\hat{\sigma}\sigma_{\mathsf{L}} \approx_s \nu\tilde{\omega}.\hat{\theta}_0^{t\delta}\hat{\sigma}\sigma_{\mathsf{R}}$.

Induction step. Suppose that $\nu\tilde{\omega}.\hat{\theta}_i^\delta\hat{\sigma}\sigma_{\mathsf{L}} \approx_s \nu\tilde{\omega}.\hat{\theta}_i^{t\delta}\hat{\sigma}\sigma_{\mathsf{R}}$ for some $i \geqslant 0$. Let us show that:

$$\nu\tilde{\omega}.\hat{\theta}_{i+1}^\delta\hat{\sigma}\sigma_{\mathsf{L}} \approx_s \nu\tilde{\omega}.\hat{\theta}_{i+1}^{t\delta}\hat{\sigma}\sigma_{\mathsf{R}}.$$

We have that $dec_{\delta(i+1)} = \mathsf{dec}(U_{\delta(i+1)}, \mathsf{a_1})$ in the frame $\hat{\theta}_{i+1}^\delta\hat{\sigma}$. We distinguish cases according to the value of $\delta(i+1)$:

- If $\delta(i+1) = 1$, then, using Lemma 11 (since we are considering the decryption step of the protocol, ϕ_A must have been successful), we know that $U_k\theta_n\hat{\sigma}\sigma = \mathsf{e_k}\theta_n\hat{\sigma}\sigma$ for $k \in \{1, 2\}$ and any $\sigma \in \{\sigma_{\mathsf{L}}, \sigma_{\mathsf{R}}\}$. Thus, we have that $res_{i+1}\hat{\theta}_{i+1}^\delta\hat{\sigma}\sigma_{\mathsf{L}} =_E \mathsf{dec}(U_1, \mathsf{a_1})\hat{\theta}_{i+1}^\delta\hat{\sigma}\sigma_{\mathsf{L}} =_E \mathsf{v_1}\sigma_{\mathsf{L}}$. But, we also have the following equalities $res_{i+1}\hat{\theta}_{i+1}^{t\delta}\hat{\sigma}\sigma_{\mathsf{R}} =_E \mathsf{dec}(U_2, \mathsf{a_1})\hat{\theta}_{i+1}^{t\delta}\hat{\sigma}\sigma_{\mathsf{R}} =_E \mathsf{v_2}\sigma_{\mathsf{R}}$. According to the definition of σ_{L} and σ_{R}, we have $\mathsf{v_1}\sigma_{\mathsf{L}} =_E \mathsf{v_2}\sigma_{\mathsf{R}} =_E \mathsf{a}$. Then, using the induction hypothesis and Lemma 3 with the free term $U = \mathsf{a}$ and we conclude.
- If $\delta(i+1) = 2$, we adopt the same argument used in the previous case (replacing 1 by 2 and 2 by 1) and conclude using the induction hypothesis and Lemma 3 with the free term $U = \mathsf{b}$.
- If $\delta(i+1) = j \in [\![3, n]\!]$, then ${}^t\delta(i+1) = \delta(i+1)$ and we have $res_{i+1} = \mathsf{dec}(U_j, \mathsf{a_1})$. According to Lemma 11, we have that $U_j\theta_n\hat{\sigma}\sigma = \mathsf{penc}(N, P, Q)\theta_n\hat{\sigma}\sigma$ for free terms N, P, Q and any $\sigma \in \{\sigma_{\mathsf{L}}, \sigma_{\mathsf{R}}\}$. Thus, $res_{i+1}\hat{\theta}_{i+1}^\delta\hat{\sigma}\sigma_{\mathsf{L}} =_E \mathsf{dec}(\mathsf{penc}(N, P, Q), \mathsf{a_1})\theta_n\hat{\sigma}\sigma_{\mathsf{L}}$. We consider two subcases:

  * $Q\theta_n\hat{\sigma}\sigma_{\mathsf{L}} =_E \mathsf{pk}(\mathsf{a_1})$. Then, $res_{i+1}\hat{\theta}_{i+1}^\delta\hat{\sigma}\sigma_{\mathsf{L}} =_E N\hat{\theta}_{i+1}^\delta\hat{\sigma}\sigma_{\mathsf{L}}$ with free N. And, in that case, we also have $Q\theta_n\hat{\sigma}\sigma_{\mathsf{R}} =_E \mathsf{pk}(\mathsf{a_1})$ (using the induction hypothesis and the fact that $\mathsf{g_1} = \mathsf{pk}(\mathsf{a_1})$) and $res_{i+1}\hat{\theta}_{i+1}^{t\delta}\hat{\sigma}\sigma_{\mathsf{R}} =_E N\hat{\theta}_{i+1}^{t\delta}\hat{\sigma}\sigma_{\mathsf{R}}$ too. Thus, we conclude using Lemma 3.

- ∗ $Q\theta_n\hat{\sigma}\sigma_{\mathsf{L}} \neq_E \mathsf{pk}(\mathsf{a}_1)$. Then, there is no reduction of res_{i+1} (in both frames) and we have two possibilities. If there exists $j \in [\![1, i]\!]$ such that $res_j\hat{\theta}^\delta_j\hat{\sigma}\sigma_{\mathsf{L}} = res_{i+1}\hat{\theta}^\delta_{i+1}\hat{\sigma}\sigma_{\mathsf{L}}$, then we conclude using Lemma 3. If there is not, then since a_1 is restricted and not deducible (Lemma 9), we have that res_{i+1} is neither deducible itself, nor a subterm of $\hat{\theta}^\delta_i\hat{\sigma}\sigma_{\mathsf{L}}$ and $\hat{\theta}^{t\delta}_i\hat{\sigma}\sigma_{\mathsf{R}}$. Now since dec is a destructor in E, we use Lemma 7 to conclude.

Finally, we have that $\nu\tilde{\omega}.\hat{\theta}^\delta_n\hat{\sigma}\sigma_{\mathsf{L}} \approx_s \nu\tilde{\omega}.\hat{\theta}^{t\delta}_n\hat{\sigma}\sigma_{\mathsf{R}}$, that is, changing the notations:

$$\nu\tilde{\omega}.\theta_\delta\hat{\sigma}\sigma_{\mathsf{L}} \approx_s \nu\tilde{\omega}.\theta_{t\delta}\hat{\sigma}\sigma_{\mathsf{R}}. \qquad \square$$

7. Further corruption cases using ProVerif

In order to study further corruption cases, we have used ProVerif, one of the only tools that can analyse equivalence properties in the context of security protocols. Since ProVerif does not handle associative and commutative (AC) symbols, we had to simplify the equational theory, yielding theory E' defined by the equations of Fig. 7. The main idea behind E' is to remove associative and commutative symbols from E. All equations besides the AC equations are left unchanged except Eqs (E-5) and (E-6). Equation (E-5) states that two encryptions can be combined, This can no longer be reflected in our ProVerif model. Equation (E-6) models re-encryption. To get rid of AC symbols, we instantiate it with the keys of the protocol (a_1, a_2 and a_3), preserving the behavior of the protocol, yielding Eq. (E'-5). The fact that we can consider a simplified equational theory weakens the attacker model: some attacks relying on crafty combinations of the messages may be missed. But as shown by our study, this allows to analyse (automatically) more corruption scenario.

Since ProVerif is designed to prove equivalences between processes that differ only by terms, we also needed to use another tool, ProSwapper [28], to cope with the (non deterministic) shuffle done by the Decryption service. More precisely, we actually used their algorithm to compute directly a slightly modified process in our ProVerif specification.

$$\mathsf{fst}(\mathsf{pair}(x, y)) = x \tag{E'-1}$$

$$\mathsf{snd}(\mathsf{pair}(x, y)) = y \tag{E'-2}$$

$$\mathsf{dec}(\mathsf{penc}(x, r, \mathsf{pk}(k)), k) = x \tag{E'-3}$$

$$\mathsf{dec}(\mathsf{blind}(\mathsf{penc}(x, r, \mathsf{pk}(k)), b), k) = \mathsf{blind}(x, b) \tag{E'-4}$$

$$\mathsf{renc}(\mathsf{penc}(x, r, \mathsf{pk}(\mathsf{a}_1)), \mathsf{a}_2) = \mathsf{penc}(x, r, \mathsf{pk}(\mathsf{a}_3)) \tag{E'-5}$$

$$\mathsf{unblind}(\mathsf{blind}(x, b), b) = x \tag{E'-6}$$

$$\mathsf{checksign}(x, \mathsf{vk}(id), \mathsf{sign}(x, id)) = \mathsf{ok} \tag{E'-7}$$

$$\mathsf{checkpfk}_1(\mathsf{vk}(id), ball, \mathsf{pfk}_1(id, r, x, ball)) = \mathsf{ok} \tag{E'-8}$$

$$\text{where } ball = \mathsf{penc}(x, r, \mathsf{a}_1)$$

$$\mathsf{checkpfk}_2(\mathsf{vk}(id), ball, bball, \mathsf{pfk}_2(\mathsf{vk}(id), \mathsf{a}_2, b, ball, bball)) = \mathsf{ok} \tag{E'-9}$$

$$\text{where } ball = \mathsf{penc}(x, r, \mathsf{a}_1) \text{ and } bball = \mathsf{blind}(\mathsf{penc}(x, r, \mathsf{a}_3), b)$$

Fig. 7. Equational theory E' used in ProVerif.

Table 1

Results and performances for proving ballot secrecy using ProVerif

Corrupted Voters	0	1	2	3	4	5
Honest Players	~1s (✓)	~6s (✓)	~40s (✓)	~16m (✓)	~32h (✓)	>48h (?)
Corrupted B	~3s (✓)	~28s (✓)	~2m (✓)	~14m (✓)	~2h (✓)	>48h (?)
Corrupted R	~2s (✓)	~17s (✓)	~2m (✓)	~12m (✓)	~95m (✓)	~19h (✓)
Corrupted A	<1s (✓)	~4s (✓)	~16s (✓)	~1m (✓)	~5m (✓)	~37m (✓)
Corr. R & Corr. A	<1s (✓)	~4s (✓)	~17s (✓)	~1m (✓)	~6m (✓)	~36m (✓)

The results[1] are displayed in Table 1 and the ProVerif files corresponding to our experimentation can be found at [35]. 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 E'.

8. Conclusion

We have proposed the first formal proof in a symbolic model that the e-voting protocol used in Norway indeed satisfies ballot secrecy, even when all but two voters are corrupted and even when the voters communications channels can be eavesdropped and when the Auditor and either the Ballot box or the Receipt generator are corrupted. As expected, ballot secrecy is no longer guaranteed if both the Ballot box and the Receipt generator are corrupted. Slightly more surprisingly, the protocol is not secure either if the Decryption service is corrupted or if the Ballot box and the Auditor are corrupted, as discussed in Section 4.3. Our results in Table 2. In this table, ✓ indicates that ballot secrecy is satisfied and × shows that there is an attack. In particular, all the attacks described in Section 4.3 are displayed in the table.

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 [16] but is limited to subterm convergent theories. An implementation has recently been proposed [15] 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 [10] but tailored to privacy properties.

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 $\mathsf{a}_1, \mathsf{a}_2, \mathsf{a}_3$ (such that $\mathsf{a}_1 + \mathsf{a}_2 = \mathsf{a}_3$) need to be securely distributed among (respectively) the Ballot box, the Receipt generator and the Decryption service. Also, a table $(V, \mathsf{s}_\mathsf{V})$ containing the blinding factor for each voter needs to be securely distributed to Ballot box and a table (V, d_V) 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.

It also remains to study other security properties such as receipt-freeness, coercion-resistance, and verifiability. Receipt-freeness seems to strongly rely on whether the voter may forge a fake table of receipts or fake the message received from the receipt generator. One important feature of the Norwegian

[1] Tests made on a MacBook Pro (OSX El Capitan 10.11.6) i5 2,3 GHz with 4Go RAM, using the ProVerif 1.94p11 version.

Table 2

Results for ballot secrecy

Corrupted Voters	0	5	n
Honest Players	✓ (Theorem 2)		
Corrupted B	✓ (Theorem 1)		
Corrupted R	✓ (ProVerif)		?
Corrupted A	✓ (ProVerif)		?
Corr. D + $\star$	× (Section 4.3)		
Corr. B & Corr. R	× (Section 4.3)		
Corr. B & Corr. A	× (Section 4.3)		
Corr. R & Corr. A	✓ (ProVerif)		?

Note: $\star$ stands for any other corrupted players (B, R, A) or none.

protocol is to ensure verifiability even when the voter's computer is not trusted. Our formal model could serve as a basis, splitting further the voter's behavior from its computer.

Supplementary data

Appendix is available at: http://dx.doi.org/10.3233/JCS-15777.

Acknowledgment

The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Program (FP7/2007-2013)/ERC grant agreement n°258865, project ProSecure.

References

[1] M. Abadi and C. Fournet, Mobile values, new names, and secure communication, in: *28th ACM Symposium on Principles of Programming Languages (POPL)*, 2001.

[2] M. Arapinis, S. Bursuc and M. Ryan, Reduction of equational theories for verification of trace equivalence: Re-encryption and AC, in: *First International Conference on Principles of Security and Trust (POST'12)*, Springer, 2012.

[3] M. Arapinis, V. Cortier and S. Kremer, When are three voters enough for privacy properties? in: *Proceedings of the 21st European Symposium on Research in Computer Security (ESORICS'16)*, 2016.

[4] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò and L. Vigneron, The AVISPA tool for the automated validation of Internet security protocols and applications, in: *17th International Conference on Computer Aided Verification (CAV)*, 2005.

[5] D. Basin, J. Dreier and R. Sasse, Automated symbolic proofs of observational equivalence, in: *ACM Conference on Computer and Communications Security (CCS'15)*, 2015.

[6] J. Benaloh, M.D. Byrne, B. Eakin, P.T. Kortum, N. McBurnett, O. Pereira, P.B. Stark, D.S. Wallach, G. Fisher, J. Montoya, M. Parker and M. Winn, STAR-Vote: A secure, transparent, auditable, and reliable voting system, in: *2013 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE'13)*, 2013.

[7] 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)*, 2011.

[8] D. Bernhard, O. Pereira and B. Warinschi, How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios, in: *International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, 2012.

[9] K. Bhargavan, R. Corin, C. Fournet and E. Zalinescu, Cryptographically verified implementations for TLS, in: *15th ACM Conference on Computer and Communications Security (CCS)*, 2008.

[10] B. Blanchet, An automatic security protocol verifier based on resolution theorem proving (invited tutorial), in: *20th International Conference on Automated Deduction (CADE)*, 2005.

[11] B. Blanchet, Vérification Automatique de Protocoles Cryptographiques: Modèle Formel et Modèle Calculatoire. Mémoire d'habilitation à diriger des recherches, Université Paris-Dauphine, 2008.

[12] B. Blanchet, M. Abadi and C. Fournet, Automated verification of selected equivalences for security protocols, *Journal of Logical and Algebraic Methods in Programming* (2008).

[13] R. Chadha, Ş. Ciobâcă and S. Kremer, Automated verification of equivalence properties of cryptographic protocols, in: *21th European Symposium on Programming (ESOP)*, 2012.

[14] D. Chaum, R. Carback, J. Clark, A. Essex, S. Popoveniuc, R.L. Rivest, P.Y.A. Ryan, E. Shen, A.T. Sherman and P.L. Vora, Corrections to scantegrity II: End-to-end verifiability by voters of optical scan elections through confirmation codes, *IEEE Trans. Information Forensics and Security* **5**(1) (2010), 194. doi:10.1109/TIFS.2010.2040672.

[15] 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)*, 2011.

[16] V. Cortier and S. Delaune, A method for proving observational equivalence, in: *22nd Computer Security Foundations Symposium (CSF)*, 2009.

[17] V. Cortier and B. Smyth, Attacking and fixing helios: An analysis of ballot secrecy, *Journal of Computer Security* (2013).

[18] V. Cortier and C. Wiedling, A formal analysis of the Norwegian E-voting protocol, in: *First International Conference on Principles of Security and Trust (POST)*, 2012.

[19] R. Cramer, R. Gennaro and B. Schoenmakers, A secure and optimally efficient multi-authority election scheme, in: *International Conference on the Theory and Application of Cryptographic Techniques (EuroCrypt)*, 1997.

[20] C. Cremers, The scyther tool: Verification, falsification, and analysis of security protocols, in: *20th International Conference of Computer Aided Verification (CAV)*, 2008.

[21] S. Delaune, S. Kremer and M. Ryan, Verifying privacy-type properties of electronic voting protocols, *Journal of Computer Security* (2009).

[22] A. Fujioka, T. Okamoto and K. Ohta, A practical secret voting scheme for large scale elections, in: *Workshop on the Theory and Application of Cryptographic Techniques: Advances in Cryptology*, 1992.

[23] K. Gjøsteen, Analysis of an internet voting protocol. Cryptology ePrint Archive, Report 2010/380, 2010.

[24] K. Gjøsteen, The Norwegian internet voting protocol. Cryptology ePrint Archive, Report 2013/473, 2013.

[25] N. Government, Article of the Norwegian government on the deployment of E-voting (last seen: 08-26-16), https://www.regjeringen.no/en/aktuelt/Internet-voting-pilot-to-be-discontinued/id764300/.

[26] A. Halderman and V. Teague, The new South Wales iVote system: Security failures and verification flaws in a live online election, in: *5th International Conference on E-voting and Identity (VoteID'15)*, 2015.

[27] D. Khader and P.Y.A. Ryan, Receipt freeness of Prêt à voter provably secure. IACR Cryptology ePrint Archive, 2011:594, 2011.

[28] P. Klus, B. Smyth and M.D. Ryan, ProSwapper: Improved equivalence verifier for ProVerif (last seen: 08-26-16), 2010, http://www.bensmyth.com/proswapper.php.

[29] 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.

[30] 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)*, 2004.

[31] J. Liu, A proof of coincidence of labeled bisimilarity and observational equivalence in applied pi calculus, 2011, http://lcs.ios.ac.cn/~jliu/papers/LiuJia0608.pdf.

[32] M. Ohkubo, F. Miura, M. Abe, A. Fujioka and T. Okamoto, An improvement on a practical secret voting scheme, in: *2nd International Information Security Workshop (ISW)*, 1999.

[33] A.T. Sherman, R. Carback, D. Chaum, J. Clark, A. Essex, P.S. Herrnson, T. Mayberry, S. Popoveniuc, R.L. Rivest, E. Shen, B. Sinha and P.L. Vora, Scantegrity mock election at Takoma Park, in: *4th International Conference on Electronic Voting 2010 (EVOTE'10)*, 2010, pp. 45–61.

[34] A. Tiu and J.E. Dawson, Automating open bisimulation checking for the spi calculus, in: *3rd IEEE Computer Security Foundations Symposium (CSF)*, 2010.

[35] C. Wiedling, Source files for ProVerif code, http://people.rennes.inria.fr/Cyrille.Wiedling/Resources/resources.html.