When are three voters enough for privacy properties?

Myrto Arapinis[1], Véronique Cortier[2], and Steve Kremer[2]

[1] University of Edinburgh - Edinburgh, UK
[2] LORIA, CNRS & Inria Nancy & Université de Lorraine, France

Abstract. Protocols for secure electronic voting are of increasing societal importance. Proving rigorously their security is more challenging than many other protocols, which aim at authentication or key exchange. One of the reasons is that they need to be secure for an arbitrary number of malicious voters. In this paper we identify a class of voting protocols for which only a small number of agents needs to be considered: if there is an attack on vote privacy then there is also an attack that involves at most 3 voters (2 honest voters and 1 dishonest voter).
In the case where the protocol allows a voter to cast several votes and counts, e.g., only the last one, we also reduce the number of ballots required for an attack to 10, and under some additional hypotheses, 7 ballots. Our results are formalised and proven in a symbolic model based on the applied pi calculus. We illustrate the applicability of our results on several case studies, including different versions of Helios and Prêt-à-Voter, as well as the JCJ protocol. For some of these protocols we can use the ProVerif tool to provide the first formal proofs of privacy for an unbounded number of voters.

Acknowledgments. This work has received funding from the European Research Council (ERC) under the EU's Horizon 2020 research and innovation program (grant agreement No 645865-SPOOC) and the ANR project SEQUOIA ANR-14-CE28-0030-01.

1 Introduction

Electronic voting has been adopted in several countries, such as the United States, Estonia, Australia, Norway, Switzerland, and France, to conduct legally binding elections (or at least trials for some of them). Electronic voting systems should ensure the same properties than the traditional paper ballots systems, despite the fact that malicious users may easily intercept ballots and try to forge fake ones. One crucial property is vote privacy: no one should know how a particular voter voted. Symbolic models have been very successful in the analysis of more traditional protocols that aim at confidentiality or authentication. Many decision techniques and several tools have been developed (see [1, 2, 3] to cite only a few) which have been successfully applied to a large number of case studies including widely deployed protocols such as TLS [4]. Vote privacy in symbolic models can be expressed through a rather simple and natural property [5]: an attacker should not be able to distinguish the situation where Alice votes 0 and Bob votes 1 from the situation where the votes are swapped:

$$V_{\mathsf{Alice}}(0) \mid V_{\mathsf{Bob}}(1) \approx V_{\mathsf{Alice}}(1) \mid V_{\mathsf{Bob}}(0)$$

Despite its apparent simplicity, this property is difficult to check for several reasons. Firstly, most existing decision techniques apply to reachability properties (such as authentication and confidentiality) but not to indistinguishability properties. Another major difficulty comes from the fact that e-voting systems involve less standard cryptographic primitives and sometimes even specially designed, ad-hoc primitives (like for the protocol used in Norway [6]). Typical primitives in e-voting are homomorphic encryption, zero-knowledge proofs, reencryption mixnets, etc. Some techniques and tools [7, 8, 9, 10] for indistinguishability properties have recently been developed to automatically check indistinguishability properties and some of them can handle part of the primitives needed in e-voting. For example, ProVerif and Akiss have both been successfully applied to analyse some voting protocols [5, 10, 11, 12, 13, 14]. However, a third source of difficulty is the fact that voting systems are typically parametrized by the number of voters: both the bulletin board and the tally processes have to process as many ballots as they receive. This is typically modeled by considering processes parametrized by the number of voters. Even though parameterized protocols can be encoded in a formalism such as the applied pi calculus, such encodings are complicated and generally beyond the capabilities of what automated tools support. ProVerif, which to the best of our knowledge is the only tool that supports verification of indistinguishability properties for an unbounded number of sessions (i.e. allowing replication) generally fails to prove vote privacy. One exception is a case study of the Civitas voting system by Backes et al. [11] using ProVerif. The other tools for indistinguishability (e.g. SPEC [8], Akiss [10], and APTE [9]) can only handle a finite number of sessions. So case studies have to consider a finite number of voters [10, 12, 13, 14] unless proofs are conducted by hand [13, 15].

Contributions. Our main contribution is a reduction result for a reasonnably large class of voting protocols. If there is an attack on privacy for n voters, we show that there also exists one that only requires 3 voters: 2 honest voters are necessary to state the privacy property and then 1 dishonest is sufficient to find all existing attacks. This result significantly simplifies security proofs: there is no longer need to consider arbitrarily many voters, even in manual proofs. Moreover, this result allows the use of automated tools for checking equivalence properties and justifies previous proofs conducted for a fixed number of voters (provided at least one dishonest voter was considered).

Several protocols assume voters may revote several times. This is for example the case of Helios or Civitas. Revoting is actually crucial for coercion-resistance in Civitas. When revoting is allowed, this should be reflected in the model by letting the ballot box accept an unbounded number of ballots, and retaining only the valid ones according to the revote policy. This aspect is typically abstracted in any existing formal analysis. We show that we can simplify the analysis by reducing the total number of ballots to 10 for typical revoting policies (e.g. the last vote counts) and typical tally functions. Altogether, our result amounts in a finite model property: if there is an attack on privacy on n voters that may vote arbitrarily, then there is an attack that only requires 3 voters and at most 10 ballots. We can further reduce the number of ballots to 7 for a class of protocols that has *identifiable ballots*, that is ballots that reveal the corresponding public credentials. Of course, only 3 ballots are sufficient when revoting is disallowed.

Our result holds in a rather general setting provided that the e-voting system can be modeled as a process in the applied-pi calculus [16]. Of course, this reduction result cannot hold for *arbitrary* systems. For example, if the tally phase checks that at least 4 ballots are present then at least 4 voters are necessary to conduct an attack. So we model what we think to represent a "reasonable" class of e-voting systems. The process modeling the voter may be an arbitrary process as long as it does not depend on credentials of other voters and provided voters do not need to interact once the tally phase has started. This corresponds to the "vote and go" property, that is often desirable for practical reasons, but also excludes some protocols such as [17]. Once the vote is casted the authorities proceed as follows.

- The bulletin board (if there is one) performs only public actions such as publishing a received ballot, possibly removing some parts and possibly after some public tests, *i.e.* tests that anyone could do as well. Typical public tests are checks of signature validity, well-formedness of the ballots, or validity of zero-knowledge proofs. Alternatively, we may consider an arbitrary bulletin board in case it is corrupted since it is then part of the adversarial environment.
- Next, a revote policy is applied. We consider two particular revote policies: the policy which selects the last ballot, which is the most common one, and the policy that selects the first one, which encodes the situation where revoting is prohibited.
- Finally, the tally is computed according to some counting function. We consider in particular two very common functions: the multiset and the additive counting functions. The multiset counting function returns the votes in an arbitrary order and corresponds for example to the output of a decryption mixnet. The additive counting function returns the number of votes received by each candidate.

We believe that these conditions are general enough to capture many existing e-voting schemes.

Applications. To illustrate the applicability of our result, we re-investigate several existing analyses of e-voting protocols. First, we consider several versions of the Helios protocol [18], both in its mixnet and homomorphic versions. These versions also include the Belenios [19] protocol. We are able to use the ProVerif tool to show privacy for the mixnet versions of these protocols for a bounded number of voters and ballots. Our reduction result allows immediately to conclude that vote privacy also holds for an arbitrary number of voters. The homomorphic version of Helios is out of reach of existing tools due to the presence of associative and commutative symbols. However, our reduction result does apply, which means that the manual proof of Helios conducted in [13] did not need to consider arbitrarily many voters and could be simplified. In case one wishes to adapt this proof to Belenios [19], our reduction result would alleviate the proof. The Prêt-à-Voter [20] protocol (PaV) has been analysed using ProVerif for 2 honest voters [12]. Adding a third, dishonest, voter, we can apply our result and obtain the first proof of vote privacy for an arbitrary number of voters. Unfortunately, ProVerif did not scale up to verify automatically the protocol in presence of a dishonest voter. We were also able to apply our result (and a proof using ProVerif) to a protocol by Moran and Naor and to the JCJ protocol implemented in Civitas (without a ProVerif proof).

Related work. To our knowledge, the only other reduction result applying to voting protocols was proposed by Dreier et al. [21]. Their result states that it is sufficient to prove vote privacy for two honest voters when the protocol is observationally equivalent to a protocol consisting of the parallel composition (not sharing any secret) of a partition of the set of voters. Applicability has however only been shown to examples where this trivially holds, e.g. [17, 22] as these protocols use completely public tallying mechanisms. In general, proving the required equivalence does not seem easier than proving directly vote secrecy. Moreover, it does not apply to some well known protocols such as Helios since a dishonest voter is needed to mount the vote replay attack [13].

The results of [23,24] show how to reduce the number of agents, in the case of trace properties [23] and equivalence properties [24]. The major difference with our work is that [23, 24] simply reduce the number of *agent identities* while the number of sessions (or processes) remains the same. In contrast, we do not only reduce the number of voter identities but also the number of ballots the ballot box needs to process, yielding a simpler process.

2 Modelling security protocols

As usual in symbolic protocol analysis we model protocol messages as terms. Protocols are modelled in a process calculus, similar to the applied pi calculus [16].

2.1 Messages

We assume an infinite set of *names* $\mathcal{N} = \{a, b, k, n, \ldots\}$ (which are used to represent keys, nonces, ...) and an infinite set of *channels* $\mathcal{C}h = \{c, c_1, ch, ch_1, \ldots\}$ (which are used to represent communication channels). We also consider a set of *variables* $\mathcal{X} = \{x, y, \ldots\}$, and a signature Σ consisting of a finite set of *function symbols*.

Terms are defined as names, variables, and function symbols applied to other terms. In particular, a channel is not a term. Let $\mathsf{N} \subseteq \mathcal{N}$ and $\mathsf{X} \subseteq \mathcal{X}$, the set of terms built from N and X by applying function symbols in Σ is denoted by $\mathcal{T}(\Sigma, \mathsf{N} \cup \mathsf{X})$. We write $fv(t)$ (resp. $fn(t)$) for the set of variables (resp. names) occurring in a term t. A term is *ground* if it does not contain any variable.

Example 1. We model asymmetric encryption, signatures, and pairs by the signature

$$\Sigma_{\mathsf{aenc}} \stackrel{\text{def}}{=} \{\mathsf{aenc}/3, \mathsf{adec}/2, \mathsf{pk}/1, \mathsf{sig}/2, \mathsf{checksig}/2, \mathsf{getmsg}/1, \mathsf{vk}/1, \langle\cdot,\cdot\rangle/2, \pi_1/1, \pi_2/1\}$$

where f/i denotes that f has arity i. Consider term $t \stackrel{\text{def}}{=} \langle \mathsf{pk}(sk), \mathsf{aenc}(\mathsf{pk}(sk), r, m)\rangle$ where $sk, r, m \in \mathcal{N}$. The term t represents a pair consisting of the public key $\mathsf{pk}(sk)$ associated to the private key sk and the encryption of message m with public key $\mathsf{pk}(sk)$ using randomness r. To improve readability, we may sometimes write $\langle t_1, \ldots, t_n\rangle$ instead of $\langle t_1, \langle \ldots \langle t_{n-1}, t_n\rangle \ldots\rangle\rangle$.

We denote by $\ell = [t_1, \ldots, t_n]$ the list of terms $t_1, \ldots, t_n$ and by $t_0 :: \ell$ the list obtained by adding the term t_0 to the head of the list, i.e., $t_0 :: \ell = [t_0, t_1, \ldots, t_n]$.

Sometimes we interpret lists as multisets and we write $\ell_1 =^{\#} \ell_2$ for the equality of the multisets corresponding to these lists.

A *substitution* is a partial function from variables to terms. The substitution σ that maps x_i to t_i $(1 \leq i \leq n)$ is denoted $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ and we write $\mathrm{dom}(\sigma) = \{x_1, \ldots, x_n\}$ for the domain of σ. We denote by $\emptyset$ the substitution whose domain is empty. We always suppose that substitutions are acyclic. As usual we extend substitutions to terms and write $t\sigma$ for the application of σ to term t.

To model algebraic properties of cryptographic primitives, we define an *equational theory* by a finite set E of equations $u = v$ with $u, v \in \mathcal{T}(\Sigma, \mathcal{X})$. We define $=_{\mathsf{E}}$ to be the smallest equivalence relation on terms, that contains E and that is closed under application of function symbols and substitutions of terms for variables.

Example 2. Continuing Example 1 we define the equational theory $\mathsf{E}_{\mathsf{aenc}}$ by the following equations.

$$\mathsf{adec}(x_k, \mathsf{aenc}(\mathsf{pk}(x_k), x_r, x_m)) = x_m \qquad \mathsf{checksig}(\mathsf{sig}(x, y), \mathsf{vk}(y)) = \mathsf{ok}$$
$$\pi_i(\langle x_1, x_2 \rangle) = x_i \quad (i \in \{1, 2\}) \qquad \mathsf{getmsg}(\mathsf{sig}(x, y)) = x$$

Then we have that $\mathsf{adec}(sk, \pi_2(t)) =_{\mathsf{E}_{\mathsf{aenc}}} m$.

To illustrate our calculus we consider the Helios e-voting protocol as running example. The Helios protocol relies on zero knowledge proofs. We next specify the equational theory for the particular zero knowledge proofs built by the Helios participants.

Example 3. The Helios zero knowledge proofs can be modelled by the signature

$$\Sigma_{\mathsf{zkp}} \stackrel{\text{def}}{=} \{\mathsf{zkp}_{\mathsf{E}}/3, \mathsf{checkzkp}_{\mathsf{E}}/2, \mathsf{okzkp}_{\mathsf{E}}/0\} \cup \{\mathsf{zkp}^m_{\mathsf{DM}}/3, \mathsf{checkzkp}^m_{\mathsf{DM}}/3, \mathsf{okzkp}^m_{\mathsf{DM}}/0\}_{m \in \mathbb{N}}$$

In case of homomorphic tally, the voters should also prove that their vote is valid, which can be modeled in a similar way. When submitting an encrypted vote, voters are required to prove that the encryption is well-formed, that is to say, that they know the corresponding plaintext and randomness. This is reflected by the following equation.

$$\mathsf{checkzkp}_{\mathsf{E}}(\mathsf{zkp}_{\mathsf{E}}(xr, xv, \mathsf{aenc}(xpk, xr, xv)), \mathsf{aenc}(xpk, xr, xv)) = \mathsf{okzkp}_{\mathsf{E}}.$$

In the decryption mixnets-based variant of the Helios protocol, the talliers output a zero knowledge proof of correct mix and decryption. Such a proof establishes that the output of the decryption mixnet is indeed a permutation of the content of the encrypted ballots received as input. This is captured by the following infinite set of equations. For all $m \in \mathbb{N}$, and all $\{i_1, \ldots, i_m\} = \{1, \ldots, m\}$,

$$\mathsf{checkzkp}^m_{\mathsf{DM}}(\mathsf{zkp}^m_{\mathsf{DM}}(xk, xciph, xplain), xciph, xplain) = \mathsf{okzkp}^m_{\mathsf{DM}}$$

with $xciph = (\mathsf{aenc}(\mathsf{pub}(xk), xr_1, xv_1), \ldots, \mathsf{aenc}(\mathsf{pub}(xk), xr_m, xv_m))$ and $xplain = (xv_{i_1}, \ldots, xv_{i_m})$.

In all the examples of this section, we will consider the signature $\Sigma = \Sigma_{\mathsf{aenc}} \cup \Sigma_{\mathsf{zkp}}$ and the equational theory $\mathsf{E} = \mathsf{E}_{\mathsf{aenc}} \cup \mathsf{E}_{\mathsf{zkp}}$.

We say that a symbol $+$ is associative and commutative (AC in short) w.r.t. an equational theory E if E contains the two equations:

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

2.2 Processes

We model protocols using a process calculus. Our *plain processes* are similar to plain processes in applied pi calculus [16] and are defined through the grammar given in Figure 1 where c is a channel, t, t_1, t_2 are terms, x is a variable, n is either a name or a channel, and $i \in \mathbb{N}$ is an integer. The terms t, t_1, t_2 may contain variables.

The process 0 does nothing. $P \mid Q$ behaves as the parallel execution of processes P and Q. $\nu n.P$ restricts the scope of n. When n is a name, it typically represents a freshly generated, secret value, *e.g.*, a key or a nonce, in P. When n is a channel, it declares a private channel, that cannot be accessed by the adversary. Replication $!P$ behaves as an unbounded number of copies of P. The conditional if $t_1 = t_2$ then P else Q behaves as P if t_1 and t_2 are equal in the equational theory and as Q otherwise. The process $c(x).P$ inputs a message t on channel c, binds it to x and then behaves as P where x has been replaced by t. $\overline{c}\langle t\rangle.Q$ outputs message t on channel c before behaving as Q. Our calculus also introduces a *phase* instruction, in the spirit of [24, 25], denoted $i : P$. We denote by $Phase(P)$ the set of phases that appears in P, that is the set of j such that $j : Q$ occurs in P. By a slight abuse of notations, we write $Phase(P) < Phase(Q)$ if any phase in $Phase(P)$ is smaller than any phase in $Phase(Q)$.

$$
\begin{array}{rl}
P, Q := & 0 \\
 & P \mid Q \\
 & \nu n.P \\
 & !P \\
 & \text{if } t_1 = t_2 \text{ then } P \text{ else } Q \\
 & c(x).P \\
 & \overline{c}\langle t\rangle.Q \\
 & i : P
\end{array}
$$

Fig. 1. Syntax of plain processes

As usual, names and variables have scopes, which are delimited by restrictions and inputs. We write $fv(P)$, $bv(P)$, $fn(P)$ and $bn(P)$ for the sets of free and bound variables, and free and bound names of a plain process P respectively.

Example 4. A voter in Helios proceeds as follows. She computes her ballot by encrypting her vote with the public key $\mathsf{pk}(skE)$ of the election. The corresponding secret key is shared among several election authorities, which is not modeled here. Then she casts her ballot together with her identity and a zero knowledge proof through an authenticated channel. All this information will be published on a public bulletin board. The process $V(\mathsf{pk}(skE), cred, id, v)$ models the actions of a voter with identity id and credential $cred$ casting a ballot for candidate v:

$$V(\mathsf{pk}(skE), cred, id, v) \stackrel{\text{def}}{=} \nu r.\ \overline{bb}\langle\langle id, \mathsf{sig}(bal, cred), prf\rangle\rangle$$

where $bal = \mathsf{aenc}(\mathsf{pk}(skE), r, v)$ and $prf = \mathsf{zkp}_{\mathsf{E}}(r, v, bal)$. The authenticated channel is modelled by a signature although Helios relies on a login/password mechanism.

Extended processes keep track of additional information during an execution: the names that have been bound, the currently active processes that are running in parallel, the history of messages that were output by the process and the current phase.

Definition 1 (Extended process). *An* extended process *is a tuple* $(\mathcal{E}; \mathcal{P}; \Phi; i)$ *where:*

- *$\mathcal{E}$ is a set of names and channels that are restricted in $\mathcal{P}$ and Φ;*
- *$\mathcal{P}$ is a multiset of* plain processes *with $fv(\mathcal{P}) = \emptyset$;*
- *$\Phi = \{x_1 \mapsto u_1, \ldots, x_n \mapsto u_n\}$ is a ground substitution where $u_1, \ldots, u_n$ represent the messages previously output to the environment.*
- *i is an integer denoting the current phase.*

Example 5. The following extended process models two honest Helios voters id_A and id_B ready to cast their ballots v_A and v_B respectively in a first phase, and the Helios tallying authorities Tal ready to tally the cast ballots in a second phase

$$\mathsf{Helios}(v_A, v_B) \stackrel{\text{def}}{=} (\mathcal{E}_0, 1 : V_A \mid 1 : V_B \mid 2 : Tal, \emptyset, 1)$$

where $\mathcal{E}_0$ is a set of names with $cred_A, cred_B \in \mathcal{E}_0$,

$$V_A \stackrel{\text{def}}{=} V(\mathsf{pk}(skE), cred_A, id_A, v_A) \text{ and } V_B \stackrel{\text{def}}{=} V(\mathsf{pk}(skE), cred_B, id_B, v_B)$$

model the two honest voters where V is defined in Example 4, and

$$Tal \stackrel{\text{def}}{=} bb(xb_A).bb(xb_B).T$$

for some process T modelling the tallying authorities.

Given $A = (\mathcal{E}; \mathcal{P}; \Phi; i)$, we define the set of free and bound names of A as $fn(A) = (fn(\mathcal{P}) \cup fn(\Phi)) \setminus \mathcal{E}$, and $bn(A) = bn(\mathcal{P}) \cup \mathcal{E}$. Similarly free and bound variables are defined as $fv(A) = (fv(\mathcal{P}) \cup \mathrm{dom}(\Phi))$, and $bv(A) = bv(\mathcal{P})$. An extended process A is closed if $fv(A) = \mathrm{dom}(\Phi)$.

The operational semantics of our calculus is defined by a labelled transition system which allows to reason about processes that interact with their environment. The transition relation $A \xrightarrow{\ell} B$ relates two ground extended processes A and B and is decorated by a label ℓ, which is either an input ($c(M)$), an output ($\nu x.\overline{c}\langle x\rangle$), or a silent action ($\tau$). Silent actions are standard, while visible input and output actions are interactions with the adversary on public channels. An output label $\nu x.\overline{c}\langle x\rangle$ reflects that messages are output "by reference": the label contains the variable added to $\mathrm{dom}(\Phi)$ which maps to the ground message that was output. The input label $c(M)$ contains the term M used by the adversary to compute the message: M may be constructed from previous outputs (adressed through variables in $\mathrm{dom}(\Phi)$), but is not allowed to use private names. The precise definition of the transition relation is given in Appendix **??**.

Notations. Given a set $\mathcal{S}$ we denote by $\mathcal{S}^*$ the set of all finite *sequences* of elements in $\mathcal{S}$. We may also write $\tilde{u}$ for the finite sequence $u_1, \ldots, u_n$. Let $\mathcal{A}$ be the alphabet of actions (in our case this alphabet is infinite and contains the special symbol τ). For every $w \in \mathcal{A}^*$, the relation $\xrightarrow{w}$ on processes is defined in the usual way, *i.e.*, we write $A \xrightarrow{w} A'$ when $w = \ell_1 \ell_2 \cdots \ell_n$ and $A \xrightarrow{\ell_1} A_1 \xrightarrow{\ell_2} \ldots \xrightarrow{\ell_n} A'$. For $s \in (\mathcal{A} \setminus \{\tau\})^*$, the relation $\stackrel{s}{\Rightarrow}$ on processes is defined by: $A \stackrel{s}{\Rightarrow} B$ if, and only if there exists $w \in \mathcal{A}^*$ such that $A \xrightarrow{w} B$ and s is obtained by erasing all occurrences of τ from w.

Example 6. Continuing our running example we illustrate the operational semantics by the following transitions

$$Helios(v_A, v_B) \xrightarrow{\nu y_A.\overline{bb}\langle y_A\rangle} \xrightarrow{\nu y_B.\overline{bb}\langle y_B\rangle} \xrightarrow{\mathsf{phase}\ 2} (\mathcal{E}; T; \Phi; 2) \quad \text{where}$$

- $\mathcal{E} = \mathcal{E}_0 \cup \{r_A, r_B\}$,
- $\Phi = \{y_A \mapsto \langle id_A, \mathsf{sig}(bal_A, cred_A), prf_A\rangle, y_B \mapsto \langle id_B, \mathsf{sig}(bal_B, cred_B), prf_B\rangle\}$ where $bal_C = \mathsf{aenc}(\mathsf{pk}(skE), r_C, v_C)$ and $prf_C = \mathsf{zkp}_{\mathsf{E}}(r_C, v_C, bal_C)$ for $C \in \{A, B\}$.

A *frame* $\varphi = \nu\mathcal{E}.\Phi$ consists of a set of names $\mathcal{E}$ and a substitution $\Phi = \{x_1 \mapsto u_1, \ldots, x_n \mapsto u_n\}$. The names $\mathcal{E}$ are bound in φ and can be α-converted. Moreover names can be added (or removed) to (from) $\mathcal{E}$ as long as they do not appear in Φ. We write $\varphi =_\alpha \varphi'$ when frames φ and φ' are equal up to α-conversion and addition/removal of unused names. In this way two frames can always be rewritten to have the same set of bound names. When $A = (\mathcal{E}; \mathcal{P}; \Phi; i)$ is an extended process, we define $\phi(A) \stackrel{\text{def}}{=} \nu\mathcal{E}.\Phi$.

Given a frame $\varphi = \nu\mathcal{E}.\Phi$ an attacker can construct new terms building on the terms exposed by φ. For this the attacker applies a *recipe* on the frame. A recipe R for a frame φ is any term such that $fn(R) \cap \mathcal{E} = \emptyset$ and $fv(R) \subseteq \mathrm{dom}(\Phi)$. An attacker is unable to distinguish two sequences of messages if he cannot construct a test that distinguishes them. This notion is formally captured by *static equivalence* [16] of frames.

Definition 2 (Static equivalence). *Two frames $\varphi_1 =_\alpha \nu\mathcal{E}.\Phi_1$ and $\varphi_2 =_\alpha \nu\mathcal{E}.\Phi_2$ are statically equivalent, noted $\varphi_1 \sim \varphi_2$ when $\mathrm{dom}(\Phi_1) = \mathrm{dom}(\Phi_2)$, and for all recipes M and N of φ_1 we have that $M\Phi_1 =_{\mathsf{E}} N\Phi_1$ iff $M\Phi_2 =_{\mathsf{E}} N\Phi_2$.*

Note that in the above definition the frames φ_1 and φ_2 have the same set of recipes as they bind the same names $\mathcal{E}$ and their substitutions have the same domain.

Example 7. Let Φ be the substitution of Example 6 and

$$\Phi' = \{y_A \mapsto \langle id_A, \mathsf{sig}(bal'_A, cred_A), prf'_A\rangle, y_B \mapsto \langle id_B, \mathsf{sig}(bal'_B, cred_B), prf'_B\rangle\}$$

where $bal'_C = \mathsf{aenc}(\mathsf{pk}(skE), r_C, v_D)$ and $prf'_C = \mathsf{zkp}_{\mathsf{E}}(r_C, v_D, bal'_C)$ for $C, D \in \{A, B\}$ with $C \neq D$. Since $\mathsf{adec}(skE, \pi_1(\pi_1(\mathsf{getmsg}(y_A))))\Phi =_{\mathsf{E}} v_A$, but $\mathsf{adec}(skE, \pi_1(\pi_1(\mathsf{getmsg}(y_A))))\Phi' \neq_{\mathsf{E}} v_A$, we have that

$$\nu skE.\nu r_A.\nu r_B.\Phi \sim_{\mathsf{E}} \nu skE.\nu r_A.\nu r_B.\Phi' \text{ while } \nu r_A.\nu r_B.\Phi \not\sim_{\mathsf{E}} \nu r_A.\nu r_B.\Phi'$$

Indeed, an attacker may distinguish between these two frames as soon as he has the secret key skE, by simply decrypting the ballots.

Given two extended processes A_1 and A_2, we often write $A_1 \sim A_2$ for $\phi(A_1) \sim \phi(A_2)$. Given an extended process A we define its set of traces as

$$\mathsf{traces}(A) \stackrel{\text{def}}{=} \{(tr, B) \mid A \stackrel{tr}{\Rightarrow} B\}$$

We can now define what it means for an attacker to be unable to *distinguish* two processes even if he is allowed to actively interact with them. This notion of indistinguishability is naturally modelled by *trace equivalence*.

Definition 3 (Trace equivalence). *Let A and B be two closed extended processes. A is trace included in B, written $A \sqsubseteq B$, if for every trace $(tr, A') \in \mathsf{traces}(A)$ there exists B' such that $(tr, B') \in \mathsf{traces}(B)$ and $A' \sim B'$. A and B are* trace equivalent*, denoted $A \approx B$, if $A \sqsubseteq B$ and $B \sqsubseteq A$.*

Intuitively, as the sequence of visible actions in the labels encode the adversary's actions the definition requires that for the same interaction with the adversary the protocols produce indistinguishable outputs.

3 Modelling e-voting protocols

In this section we explain how we formally model e-voting protocols and state the assumptions needed for our results.

Since many e-voting protocols use zero-knowledge proofs, we consider a signature Σ with $\mathsf{zkp}, \mathsf{checkzkp}, \mathsf{okzkp} \in \Sigma$ and we assume an equational theory that can be described by an AC-convergent (possibly infinite) rewrite theory such that the only rules in which zkp, $\mathsf{checkzkp}$, and okzkp occur, are of the form:

$$\mathsf{checkzkp}(\mathsf{zkp}(U_1, \ldots, U_m), V_1, \ldots, V_n) \to \mathsf{okzkp}$$

where $\mathsf{zkp}, \mathsf{checkzkp}, \mathsf{okzkp}$ do not occur in the U_i, V_j. Since the terms U_i, V_j are left unspecified, this captures most existing zero-knowledge proofs. In particular, it covers the zero-knowledge proofs considered in Example 3.
A voting protocol is a family of processes $\{\Pi^{n_h,n_d,m}(\mathcal{C}r^h_{n_h}, \mathcal{C}r^d_{n_d}, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}})\}_{n_h,n_d,m\in\mathbb{N}}$ where

- n_h and n_d are the number of honest and dishonest voters respectively;
- $\mathcal{C}r^h_{n_h}$ (*resp.* $\mathcal{C}r^d_{n_d}$) is the set of n_h (*resp.* n_d) voting credentials which determines the set of honest eligible voters (*resp.* dishonest eligible voters), such that $\mathcal{C}r^h_{n_h} \cap \mathcal{C}r^d_{n_d} = \emptyset$. Each credential $\tilde{cr} \in \mathcal{C}r^h_{n_h} \cup \mathcal{C}r^d_{n_d}$ is a sequence of terms;
- m is the number of ballots accepted during the tally;
- $\mathcal{K}_{\mathsf{pv}}$ (*resp.* $\mathcal{K}_{\mathsf{pb}}$) is the set of all private (*resp.* public) material.

As usual it is sufficient to consider voting processes that model only the honest voters and the tally (the dishonest voters are left unspecified as part of the environment, and their credentials are public). We may assume w.l.o.g. that the tally process starts with a fresh phase and first reads the ballots on the board. Formally, we assume that voting processes are of the form:

$$\Pi^{n_h,n_d,m}(\mathcal{C}r^h_{n_h}, \mathcal{C}r^d_{n_d}, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}) \stackrel{\text{def}}{=} V(\tilde{cr}_1) \mid V(\tilde{cr}_2) \mid \cdots \mid V(\tilde{cr_{n_h}}) \mid \mathsf{tall} : bb(x_1).\ \ldots\ .bb(x_m).T^{n,m}(\mathcal{C}r_n, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}})$$

where $\mathcal{C}r_n = \mathcal{C}r^h_{n_h} \cup \mathcal{C}r^d_{n_d}$, and for all $i \in \{1, \ldots, n_h\}$, $\tilde{cr}_i \in \mathcal{C}r^h_{n_h}$. Furthermore, we require that $Phase(V) < \mathsf{tall}$, $Phase(T^{n,m}) = \emptyset$ and $T^{n,m}(\mathcal{C}r_n, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}})$ contains at most one output which is performed on the channel tal. We note that from the above structure of a voting process it follows that all traces are prefixes of traces of the form

$$tr' \cdot \mathsf{phase}\ \mathsf{tall} \cdot bb(RB_1) \ldots bb(RB_m) \cdot \nu y.\overline{tal}\langle y \rangle.$$

$V(\tilde{cr})$ models an honest voter, whose credentials are $\tilde{cr}$. $T^{n,m}(\mathcal{C}r_n, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}})$ is the remainder of the tallier process. It is parameterised by the number m of ballots it accepts and the number n of eligible voters. We require that $V(\tilde{cr})$ be independent of n and m and does not use any other credentials, i.e. $fn(V(\tilde{cr})) \cap \mathcal{C}r_n \subseteq \{\tilde{cr}\}$. These are the only restrictions on the voter process and we believe them to be reasonable and natural.

An e-voting protocol proceeds in two phases: vote casting and tallying. During the vote phase all voters simply cast their ballots. The tally phase proceeds as follows. First

m ballots are input. Then a *public test* is applied to these ballots to carry out a first validity check, e.g. verify some zero knowledge proofs ensuring that the ballots are well formed. Next, the *revote policy* is applied to remove votes cast by a same voter, e.g., keep only the last one. Finally, the process performs the tally and outputs the result.

3.1 Public tests

As explained above, the ballot box may apply public tests to the casted ballots. Public tests are Boolean combinations over atomic formulas of the form $M = N$ where $M, N \in \mathcal{T}(\Sigma, \mathsf{X})$, *i.e.* they do not contain any names. An atomic formula is satisfied when $M =_{\mathsf{E}} N$ and we lift satisfaction to tests as expected.

We assume a family of tests $\{\mathsf{Test}^m\}_{m\in\mathbb{N}}$ where m is the number of casted ballots that are tested and Test^m contains m distinguished variables $x_1, \ldots, x_m$ to be substituted by the ballots. We write $\mathsf{Test}^m([B_1, \ldots, B_m]) = \top$ when the test $\mathsf{Test}^m\{x_1 \mapsto B_1, \ldots, x_m \mapsto B_m\}$ is satisfied. Finally we say that a *test is voting-friendly* whenever satisfaction is preserved on sublists of ballots, that is $\mathsf{Test}^m([B_1, \ldots, B_m]) = \top$ implies $\mathsf{Test}^h([B_{i_1}, \ldots, B_{i_h}]) = \top$ for any $1 \leq i_1 < \cdots < i_h \leq m$.

We believe this condition to be natural. It discards contrived tests that would accept a ballot only if another ballot is present. Conversely, we may consider tests that discard lists with duplicate ballots.

Example 8. The public test applied by the tallying authorities in the Helios protocol consists of two parts. First, a local test that checks the zero knowledge proofs of each submitted ballot, and second, a global test that checks that encrypted votes are pairwise distinct. This is to avoid the replay attack mentioned in [13]. Such checks are formally reflected by the family of tests $\{\mathsf{Test}^m\}_{m\in\mathbb{N}}$ with

$$Test^m([B_1, \ldots, B_m]) \stackrel{\text{def}}{=} \bigwedge_{i=1}^{i=m} \mathsf{lTest}(B_i) \bigwedge_{i,j\in\{1,\ldots,m\}}^{i\neq j} \mathsf{gTest}(B_i, B_j)$$

$$\mathsf{lTest}(B) \stackrel{\text{def}}{=} \begin{cases} \top \text{ if } B = \langle id, bal, prf\rangle \text{ and } \mathsf{checkzkp_E}(\mathsf{getmsg}(bal), prf) =_{\mathsf{E}} \mathsf{okzkp_E} \\ \bot \text{ otherwise} \end{cases}$$

$$\mathsf{gTest}(B, B') \stackrel{\text{def}}{=} \begin{cases} \top \text{ if } B = \langle id, bal, prf\rangle \text{ and } B' = \langle id', bal', prf'\rangle \\ \quad \text{and } \mathsf{getmsg}(bal) \neq \mathsf{getmsg}(bal') \\ \bot \text{ otherwise} \end{cases}$$

3.2 Revote policies

Many e-voting protocols offer voters the possibility to cast several votes, keeping eventually only one vote per voter, e.g. the last submitted ballot. Which vote is kept depends on the particular policy. Re-voting intends to guarantee some protection against coercion. We formalize the notion of policy as a function $\mathsf{Policy}^{n,m}$ which takes a list of m terms (intuitively, the vote and credential) and a set of n credentials (honest and dishonest) and returns the sublist of selected terms to be tallied. A protocol will depend on a family of such policy functions $\{\mathsf{Policy}^{n,m}\}_{n,m\in\mathbb{N}}$. We consider two particular, but standard revote policies. The most usual one selects the last cast vote:

$$\mathsf{Policy}^{n,m}_{\mathsf{last}}([V_1, \ldots, V_m], \mathcal{C}r_n) \stackrel{\text{def}}{=} [V_{i_1}, \ldots, V_{i_k}]$$

where each $V_{i_j} = (v, \tilde{cr})$ is the last occurence of the credential $\tilde{cr} \in \mathcal{C}r_n$ in the list $[V_1, \ldots, V_m]$. We also consider the policy which only keeps the first vote of each voter:

$$\mathsf{Policy}^{n,m}_{\mathsf{first}}([V_1, \ldots, V_m], \mathcal{C}r_n) \stackrel{\text{def}}{=} [V_{i_1}, \ldots, V_{i_k}]$$

where each $V_{i_j} = (v, \tilde{cr})$ is the first occurence of the credential $\tilde{cr} \in \mathcal{C}r_n$ in the list $[V_1, \ldots, V_m]$. Such a policy typically models the norevote policy (a voter cannot revote).

3.3 Extracting ballots and counting votes

A voting protocol should tally the ballots "as expected". Formally, what is expected can be formalized through an *extract* and a *counting* function.

Given a ballot B, and two sets of terms $\mathcal{K}_{\mathsf{pb}}$ and $\mathcal{K}_{\mathsf{pv}}$ representing the public and private material, the extraction function $\mathsf{Extract}$ returns the corresponding vote and credential, or $\bot$ when a ballot is not well formed., i.e., $\mathsf{Extract}(B, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}) \in (\mathcal{V} \times \mathcal{C}r_n) \cup \{\bot\}$. Moreover, we lift the extract function to lists of m ballots by applying the function pointwise, i.e., $\mathsf{Extract}^m([B_1, \ldots, B_m], \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}) \stackrel{\text{def}}{=}$

$$[\mathsf{Extract}(B_1, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}), \ldots, \mathsf{Extract}(B_m, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}})]$$

Similar extract functions have been introduced in [26] to define ballot privacy.

Example 9. The $\mathsf{Extract}$ function for the Helios protocol decrypts the encrypted vote and associates it with the signature associated to the ballot:

$$\mathsf{Extract}(B, \{skE\}, \{\mathsf{pk}(skE)\}) \stackrel{\text{def}}{=}$$

$$\begin{cases} (v, (id, cred)) & \text{if } B = \langle id, bal, prf \rangle \text{ and } bal =_{\mathsf{E}} \mathsf{sig}(\mathsf{aenc}(\mathsf{pk}(skE), r, v), cred) \\ \bot & \text{otherwise} \end{cases}$$

Similarly the counting function defines how the protocol is supposed to tally the votes. The function Count^ℓ takes as input a list of ℓ pairs $(v, cr) \in \mathcal{V} \times \mathcal{C}r$ and returns a list of terms as the election result.

Definition 4. *Let $\{\mathsf{Count}^\ell\}_{\ell \in \mathbb{N}}$ be a family of counting functions. $\{\mathsf{Count}^\ell\}_{\ell \in \mathbb{N}}$ is* voting-friendly *if for all m, n and lists of terms W_1 of size m, W_2 of size n we have that*

1. *if $W_1 =^{\#} W_2$ then $\mathsf{Count}^m(W_1) =^{\#} \mathsf{Count}^n(W_2)$;*
2. *if $\mathsf{Count}^m(W_1) =^{\#} \mathsf{Count}^n(W_2)$*
   then $\mathsf{Count}^{m+1}((v_1, cr_1) :: W_1) =^{\#} \mathsf{Count}^{n+1}((v_2, cr_2) :: W_2)$ iff $v_1 = v_2$

The first assumption requires that the result does not depend on the order in which votes are provided (intuitively, only valid votes are kept at this stage). We believe this property to be natural and it excludes contrived counting functions that would, e.g., only keep votes at even positions. The second assumption states that we may count "step by step". This is more restrictive since it excludes the majority function, i.e., the function that only outputs the name of the candidate that received most votes. But, it captures the most common result functions, namely the multiset and the additive counting functions.

Example 10. The multiset counting function typically arises in mixnet based tallies, which simply output the list of votes (intuitively once votes have been shuffled).

$$\mathsf{Count}^1_{\mathsf{Mix}}([V_1]) \stackrel{\text{def}}{=} [v] \text{ and } \mathsf{Count}^m_{\mathsf{Mix}}([V_1, \ldots, V_m]) \stackrel{\text{def}}{=} v :: Count^{m-1}_{\mathsf{Mix}}([V_2, \ldots, V_m])$$

where $V_1 = (v, \tilde{cr})$ and $m > 1$. The additive counting function can be defined similarly. For simplicity consider a binary vote, where we just want to count the number of 1's:

$$\mathsf{Count}^1_{\mathsf{HE}}([V_1]) \stackrel{\text{def}}{=} v \text{ and } \mathsf{Count}^m_{\mathsf{HE}}([V_1, \ldots, V_m]) \stackrel{\text{def}}{=} v + Count^{m-1}_{\mathsf{HE}}([V_2, \ldots, V_m])$$

where $V_1 = (v, \tilde{cr})$, $m > 1$ and $+$ is an AC symbol. Both functions are voting-friendly.

3.4 Properties

When verifying security properties of e-voting protocols it is common to only consider processes whose runs satisfy a particular property. For instance, vote secrecy is typically expressed as the indistinguishability of two processes modelling the situations where two honest voters swap their votes. We need however to ensure that these two honest voters have indeed cast their votes successfully to avoid trivial attacks. Indeed, in a run where the attacker blocks one of these voters, but not the other, the election result will be different and the two processes would be distinguished. Therefore when checking vote secrecy one typically adds a check that guarantees that the two honest votes are counted. We simply require that a *check* $\mathsf{check}([b_1, \ldots, b_m])$ applied to a list ballots $[b_1, \ldots, b_m]$ satisfies the two following requirements:

- If $\mathsf{check}([b_1, \ldots, b_m])$ holds then we can identify two (intuitively honest) ballots b_{i_1}, b_{i_2} such that check holds for any sublist containing b_{i_1} and b_{i_2}.
- If $\mathsf{check}([b_1, \ldots, b_m])$ does not hold then it does not hold either for any sublist of these ballots or if some ballots are replaced by invalid ones (that is replaced by $\bot$).

How such a check is implemented is left unspecified, it could be by listening to private channels, successively checking signatures, etc.

3.5 E-voting processes

As often when considering trace equivalence (e.g. [10, 24]), we assume processes to be deterministic. More precisely, we require the vote phase to be determinate: if the same sequence of labels leads to two different processes then the two resulting frames have to be statically equivalent. This typically holds for standard voting processes since the voter's behaviour is deterministic. For the tallying phase we slightly relax this notion and require what we call *almost determinate*. This relaxed notion only requires that there exists an output of a tally (among all possible outputs, as the particular tally may be chosen non-deterministically) that ensures static equivalence. This allows us to capture some non-deterministic behaviors such as mixnet tally.

Definition 5. *An e-voting protocol* $\{\Pi^{n_h,n_d,m}(\mathcal{C}r^h_{n_h}, \mathcal{C}r^d_{n_d}, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}})\}_{n_h,n_d,m\in\mathbb{N}}$ *is* almost determinate *if for any set of names* $\mathcal{E}_0$*, any initial attacker knowledge* Φ_0*, any*

$m, n_h, n_d \in \mathbb{N}$, *and any traces* $(tr, A_1), (tr, A_2) \in \mathsf{traces}(\mathcal{E}_0, \Pi^{n_h,n_d,m}(\mathcal{C}r^h_{n_h}, \mathcal{C}r^d_{n_d}, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}), \Phi_0, 0)$ *we have that*

$$\forall A'_1.\ A_1 \xRightarrow{\nu x.\overline{tal}\langle x\rangle} A'_1 \ \Rightarrow \exists A'_2.\ A_2 \xRightarrow{\nu x.\overline{tal}\langle x\rangle} A'_2 \text{ and } A'_1 \sim A'_2$$

We can now put all the pieces together and link e-voting protocols to the notions of public tests, revote policies, extraction and counting functions and properties.

Definition 6. *An e-voting protocol* $\{\Pi^{n_h,n_d,m}(\mathcal{C}r^h_{n_h}, \mathcal{C}r^d_{n_d}, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}})\}_{n_h,n_d,m\in\mathbb{N}}$ *is* voting friendly *w.r.t.* check, $\{\mathsf{Test}^m\}_{m\in\mathbb{N}}$, $\{\mathsf{Policy}^{n,m}\}_{n,m\in\mathbb{N}}$, $\mathsf{Extract}$, $\{\mathsf{Count}^\ell\}_{\ell\in\mathbb{N}}$ *if it is almost determinate, if* $\{\mathsf{Test}^m\}_{m\in\mathbb{N}}$, $\{\mathsf{Policy}^{n,m}\}_{n,m\in\mathbb{N}}$, $\mathsf{Extract}$, *are voting-friendly, and if for any set of names* $\mathcal{E}_0$*, any initial attacker knowledge* Φ_0*, any* m, n_h, n_d*, and any trace* $(tr'\cdot\nu x.\mathsf{phase}\ \mathsf{tall}.bb(RB_1)\ldots bb(RB_m), A_1)$ *of* $(\mathcal{E}_0, \Pi^{n_h,n_d,m}(\mathcal{C}r^h_{n_h}, \mathcal{C}r^d_{n_d}, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}), \Phi_0, 0)$*, the resulting list of ballots* $BB = [B_1, \ldots, B_m]$ *(where* $B_i = RB_i\phi(A_1)$*) satisfies the following properties.*

1) *The tally is successful (that is* $(\nu y.\overline{tal}\langle y\rangle, A_2) \in \mathsf{traces}(A_1)$*) if and only if* BB *passes the test and the check (*$\mathsf{Test}^m(BB) = \top$ *and* $\mathsf{check}(BB) = \top$*)*

2) *Whenever the tally produces an output (that is* $(\nu y.\overline{tal}\langle y\rangle, A_2) \in \mathsf{traces}(A_1)$*) then it outputs a triple* $y\phi(A_2) = \langle res, nvotes, zkp\rangle$ *where*

- res *is the result computed by counting the votes once the extraction function and the revote policy have been applied on the bulletin board;*
- $nvotes$ *is the number of votes that has been counted;*
- zkp *is a (valid) zero-knowledge proof that would not be valid for any other list of ballots different from* BB*;*
- *either* res *is the only result the tally can produce from* BB *(typically in the homomorphic case) or the tally can produce any permutation of it (typically in the mixnet case).*

A fully formal definition can be found in Appendix **??**. *We believe most existing protocols satisfy these requirements.*

For many protocols ballots can be associated to the public credentials that were used to cast them. This is the case for Helios and some of its variants where ballots either contain the voter identity (in the original Helios) or are signed using private credentials (in the Belenios system). As we will see in the next section we can get tighter bounds for this class of protocols. Formally we define protocols with identifiable ballots as follows.

Definition 7. *An e-voting protocol* $\{\Pi^{n_h,n_d,m}(\mathcal{C}r^h_{n_h}, \mathcal{C}r^d_{n_d}, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}})\}_{n_h,n_d,m\in\mathbb{N}}$ *has* identifiable ballots *if for all* $n_h, n_d, m \in \mathbb{N}$*, for any trace*

$$(tr'\cdot\nu x.\mathsf{phase}\ \mathsf{tall}.bb(RB_1)\ldots bb(RB_m)\cdot\nu y.\overline{tal}\langle y\rangle, A)$$

of $\Pi^{n_h,n_d,m}(\mathcal{C}r^h_{n_h}, \mathcal{C}r^d_{n_d}, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}))$ *there exists a recipe* R *and a variable* x *such that*

$$\forall 1 \le i \le m.\ \textit{if}\ \mathsf{Extract}([RB_i\phi(A)], \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}) = (V, \tilde{cr})\ \textit{then}\ R_i\phi(A) = \mathsf{pub}(\tilde{cr})$$

where $R_i = R\{x \mapsto RB_i\}$.

4 Main results

Throughout the section we consider two voting protocols

$$\{\Pi_i^{n_h,n_d,m}(\mathcal{C}r_{n_h}^h, \mathcal{C}r_{n_d}^d, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}})\}_{n_h,n_d,m\in\mathbb{N}}$$

for $1 \leq i \leq 2$ which are voting-friendly for check_i, $\{\mathsf{Test}^m\}_{m\in\mathbb{N}}$, $\{\mathsf{Policy}^{n,m}\}_{n,m\in\mathbb{N}}$, $\mathsf{Extract}_i^m$, $\{\mathsf{Count}_i^\ell\}_{\ell\in\mathbb{N}}$. Note that we assume the same public test for both protocols. Moreover we assume that $n_h \geq 2$ and $m \geq n_h + n_d$.

Let $\mathcal{E}_0$ be a set of names, and Φ_0 a ground substitution representing the initial attacker knowledge. $\{A_0^{n_h,n_d,m}\}_{n_h,n_d,m\in\mathbb{N}}$ and $\{B_0^{n_h,n_d,m}\}_{n_h,n_d,m\in\mathbb{N}}$ are two families of extended processes defined as follows

$$A_0^{n_h,n_d,m} \stackrel{\text{def}}{=} (\mathcal{E}_0 \cup \mathcal{C}r_{n_h}^h, \Pi_1^{n_h,n_d,m}(\mathcal{C}r_{n_h}^h, \mathcal{C}r_{n_d}^d, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}), \Phi_0, 0)\ \forall n_h, n_d, m \in \mathbb{N}$$
$$B_0^{n_h,n_d,m} \stackrel{\text{def}}{=} (\mathcal{E}_0 \cup \mathcal{C}r_{n_h}^h, \Pi_2^{n_h,n_d,m}(\mathcal{C}r_{n_h}^h, \mathcal{C}r_{n_d}^d, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}), \Phi_0, 0)\ \forall n_h, n_d, m \in \mathbb{N}$$

Our reduction results apply to equivalences of the form $A_0^{n_h,n_d,m} \approx B_0^{n_h,n_d,m}$ for all m, n_h, n_d. Vote privacy is typically modelled in this way [5]. The proofs of the results presented in this section are detailed in Appendinces **??** and **??**.

Our first result states that attacks on such equivalences require at most 3 voters.

Proposition 1. *If* $A_0^{k_h,k_d,\ell} \not\approx B_0^{k_h,k_d,\ell}$ *then* $A_0^{2,k'_d,\ell} \not\approx B_0^{2,k'_d,\ell}$ *for* $k'_d = 0$ *or* $k'_d = 1$.

Note that this case does not yet bound the number of ballots to be considered. In particular, when re-voting is allowed the attacker may a priori need to submit several ballots in order to distinguish the two processes. In other words, the ballot box is still parameterized by the number of ballots to be received. However, whenever we assume that Π_1 and Π_2 do not allow voters to revote, we can deduce immediately that 3 ballots suffice to capture any attack. More formally, we encode this situation by letting $k = \ell$ and considering the re-vote policy that only keeps the first vote of each voter.

Theorem 1. *If* $\{\mathsf{Policy}^{n,m}\}_{n,m\in\mathbb{N}} = \{\mathsf{Policy}_{\mathsf{first}}^{n,m}\}_{n,m\in\mathbb{N}}$ *and* $A_0^{k_h,k_d,k} \not\approx B_0^{k_h,k_d,k}$ *where* $k = k_h + k_d$, *then* $A_0^{2,k'_d,k'} \not\approx B_0^{2,k'_d,k'}$ *for* $k'_d = 0$ *or* $k'_d = 1$ *and* $k' = 2 + k'_d$.

Intuitively, the case where $k'_d = 0$ corresponds to the case where an attacker can distinguish the processes playing only with two honest voters. This case for instance arises when analyzing a naive protocol where each voter simply signs his vote, hence offering no anonymity at all. The case where $k'_d = 1$ corresponds to the case where the attacker computes a vote which depends on the honest votes. The above results state that an attacker does not need more then one ballot in that case. An example of such an attack is the vote copy attack on Helios described in [13]. We could actually encode any attack with 2 voters into an attack with 3 voters by letting the adversary play like a useless, honest, voter. This would require however to formalize the fact that the attacker may always simulate an honest voter, that is, the voting process.

We now consider the case where re-voting is allowed. In this case we can bound the number of ballots that need to be considered to $4 + 2k$ (for k number of voters in total).

Proposition 2. *If* $A_0^{k_h,k_d,\ell} \not\approx B_0^{k_h,k_d,\ell}$*, then there exists* $\ell_{min} \leq 4 + 2k$ *such that* $A_0^{k_h,k_d,\ell_{min}} \not\approx B_0^{k_h,k_d,\ell_{min}}$ *where* $k = k_h + k_d$*.*

Combining the reductions on the number of voters and the number of ballots we obtain the following theorem.

Theorem 2. *If* $A_0^{k_h,k_d,\ell} \not\approx B_0^{k_h,k_d,\ell}$*, then there exists* $k'_d \in \{0,1\},\ \ell_{min} \leq 4+2k$ *such that* $A_0^{2,k'_d,\ell_{min}} \not\approx B_0^{2,k'_d,\ell_{min}}$ *where* $k = 2 + k'_d$*.*

This is an immediate consequence of Propositions 1 and 2 and yields a bound of 4+2×3=10. When protocols have identifying ballots (Definition 7) we can tighten our reduction of the number of ballots: we only need to consider $4 + k$ ballots.

Corollary 1. *If* Π_1 *and* Π_2 *have identifying ballots and* $A_0^{k_h,k_d,\ell} \not\approx B_0^{k_h,k_d,\ell}$*, then* $\exists \ell_{min} \leq 4 + k.\ A_0^{k_h,k_d,\ell_{min}} \not\approx B_0^{k_h,k_d,\ell_{min}}$ *where* $k = k_h + k_d$*.*

This is a corollary of the proof of Proposition 2. With identifiable ballots, we know that the ballots selected by the revoting policy on the left and on the right hand-side are the same. Again, we combine this result with the reduction on the number of voters.

Theorem 3. *If* Π_1 *and* Π_2 *have identifying ballots and* $A_0^{k_h,k_d,\ell} \not\approx B_0^{k_h,k_d,\ell}$ *then* $\exists k'_d \in \{0,1\},\ \ell_{min} \leq 4 + k$ *such that* $A_0^{2,k'_d,\ell_{min}} \not\approx B_0^{2,k'_d,\ell_{min}}$ *where* $k = 2 + k'_d$*.*

This is an immediate consequence of Corollary 1 and Proposition 1 and yields a bound of 4+3=7 ballots.

5 Case studies

We apply our results on several case studies: several versions of Helios [18, 19, 27] and Prêt-à-Voter [20], as well as the JCJ protocol [28] implemented in the Civitas system [29]. For some of these protocols we show that the ProVerif verification tool [1] can be used to perform a security proof that, thanks to our results, is valid for an arbitrary number of voters and ballots.

For the other protocols, ProVerif is not able to verify the protocols, either due to the fact that equational theories with AC symbols are not supported by ProVerif or simply because of a state explosion problem. In these cases we show that our results nevertheless apply. Given recent progress in automated verification for equivalence properties [9,10,30] we hope that verification of some of these protocols will be possible soon. Our results would also be useful to simplify proofs by hand.

The results in this section are summarized in Figure 2. Our hypotheses were always satisfied wherever applicable. For several protocols, we could not conduct the analysis with ProVerif, either because the equational theory is out of reach of the tool or because we had to stop ProVerif execution after a couple of hours. The case studies are further detailed in Appendices **??**, **??**, and **??**. The results presented in this section rely on ProVerif scripts available at `http://3voters.gforge.inria.fr`.

	3 ballots (Theorem 1)	
	Hyp	ProVerif
PaV (DM)	✓	✓
PaV (RM)	✓	×
Helios mix (weeding)	✓	✓
Helios mix (id in zkp)	✓	✓
Helios hom (weeding)	✓	×
Helios hom (id in zkp)	✓	×
Belenios mix	✓	✓
Belenios hom	✓	×

(a) Protocols without revoting.

	7 ballots (Theorem 3)		10 ballots (Theorem 2)	
	Hyp	ProVerif	Hyp	ProVerif
Helios mix (weeding)	✓	✓	✓	×
Helios mix (id in zkp)	✓	✓	✓	×
Helios hom (weeding)	✓	×	✓	×
Helios hom (id in zkp)	✓	×	✓	×
Belenios mix	✓	✓	✓	×
Belenios hom	✓	×	✓	×
JCJ	✓	×	✓	×

(b) Protocols with revoting.

Fig. 2. Summary of application of our results on case studies. A × in the "ProVerif" column indicates that we could not successfully run the analysis with ProVerif.

6 Conclusion

In this paper we propose reduction results for e-voting protocols that apply to vote privacy. We believe they also apply to stronger properties such as receipt-freeness. Our first reduction result states that whenever there is an attack, there is also an attack with only two honest voters and at most one dishonest voter. This considerably simplifies the proofs and encodings otherwise needed to verify such protocols using automated verification tools. We moreover consider the case where the protocol allows a voter to cast multiple votes and selects one vote according to a given re-vote policy, e.g. select the last vote casted. In that case verifying privacy is still complicated even when restricted to three voters. We therefore show a second reduction result that allows to consider at most 10 ballots. In case the protocol has *identifiable ballots* we reduce the number of necessary ballots to 7. We have shown that the hypotheses of our theorems are satisfied by many protocols: several variants of Helios, Prêt-à-Voter, as well as Civitas. For several of these protocols we were able to apply automated tool verification and provide the first automated proofs for an unbounded number of voters and ballots. For the decryption mixnets-based PaV protocol, we even provide the first proof of vote privacy.

An interesting direction for future work is to further tighten the bound on the number of ballots, possibly characterizing properties enjoyed by voting protocols. We also foresee to show similar reduction results for other properties of e-voting, such as verifiability. Given that the result is stated in a symbolic model, we also plan to investigate if the result can be transposed to a computational model.

References

1. B. Blanchet, "An efficient cryptographic protocol verifier based on Prolog rules," in *14th Computer Security Foundations Workshop (CSFW'01)*. IEEE Comp. Soc., 2001, pp. 82–96.

2. M. Rusinowitch and M. Turuani, "Protocol insecurity with finite number of sessions is NP-complete," in *Proc. 14th Computer Security Foundations Workshop (CSFW'01)*. IEEE Comp. Soc., 2001, pp. 174–190.
3. H. Comon-Lundh and V. Shmatikov, "Intruder deductions, constraint solving and insecurity decision in presence of exclusive or," in *Proc. 18th Annual Symp. on Logic in Computer Science (LICS'03)*. IEEE Comp. Soc., 2003, pp. 271–280.
4. K. Bhargavan, R. Corin, C. Fournet, and E. Zalinescu, "Cryptographically verified implementations for tls," in *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS'08)*, October 2008, pp. 459–468.
5. S. Kremer and M. D. Ryan, "Analysis of an electronic voting protocol in the applied pi-calculus," in *Proc. 14th European Symposium On Programming (ESOP'05)*, ser. LNCS, vol. 3444. Springer, 2005, pp. 186–200.
6. K. Gjøsteen, "Analysis of an internet voting protocol," Cryptology ePrint Archive, Report 2010/380, 2010, http://eprint.iacr.org/.
7. B. Blanchet, M. Abadi, and C. Fournet, "Automated Verification of Selected Equivalences for Security Protocols," in *20th Symp. on Logic in Computer Science (LICS'05)*, Jun. 2005, pp. 331–340.
8. A. Tiu and J. E. Dawson, "Automating open bisimulation checking for the spi calculus," in *Proc. 23rd Computer Security Foundations Symp. (CSF'10)*. IEEE Comp. Soc., 2010, pp. 307–321.
9. V. Cheval, H. Comon-Lundh, and S. Delaune, "Trace equivalence decision: Negative tests and non-determinism," in *Proc. 18th ACM Conference on Computer and Communications Security (CCS'11)*. ACM, Oct. 2011.
10. R. Chadha, Ş. Ciobâcă, and S. Kremer, "Automated verification of equivalence properties of cryptographic protocols," in *21th European Symposium on Programming (ESOP'12)*, ser. LNCS, vol. 7211. Springer, Mar. 2012, pp. 108–127.
11. M. Backes, C. Hritcu, and M. Maffei, "Automated verification of remote electronic voting protocols in the applied pi-calculus," in *21st IEEE Computer Security Foundations Symp. (CSF'08)*. IEEE Comp. Soc., 2008, pp. 195–209.
12. M. Arapinis, S. Bursuc, and M. Ryan, "Reduction of equational theories for verification of trace equivalence: re-encryption and ac," in *First Conference on Principles of Security and Trust (POST'12)*, 2012, pp. 169–188.
13. V. Cortier and B. Smyth, "Attacking and fixing helios: An analysis of ballot secrecy," *Journal of Computer Security*, vol. 21, no. 1, pp. 89–148, 2013.
14. M. Arapinis, V. Cortier, S. Kremer, and M. D. Ryan, "Practical Everlasting Privacy," in *Proceedings of the 2nd Conferences on Principles of Security and Trust (POST'13)*, ser. LNCS, vol. 7796. Springer, Mar. 2013, pp. 21–40.
15. V. Cortier and C. Wiedling, "A formal analysis of the norwegian e-voting protocol," in *Proceedings of the 1st International Conference on Principles of Security and Trust (POST'12)*, ser. LNCS, vol. 7215. Springer, Mar. 2012, pp. 109–128.
16. M. Abadi and C. Fournet, "Mobile values, new names, and secure communication," in *Proc. 28th ACM Symp. on Principles of Programming Languages (POPL'01)*. ACM, 2001, pp. 104–115.
17. T. Okamoto, "Receipt-free electronic voting schemes for large scale elections," in *Proc. 5th Int. Security Protocols Workshop*, ser. LNCS, vol. 1361. Springer, 1997, pp. 25–35.
18. B. Adida, "Helios: web-based open-audit voting," in *17th conference on Security symposium (SS'08)*. USENIX Association, 2008, pp. 335–348. [Online]. Available: http://dl.acm.org/citation.cfm?id=1496711.1496734
19. V. Cortier, D. Galindo, S. Glondu, and M. Izabachene, "Election verifiability for helios under weaker trust assumptions," in *19th European Symp. on Research in Computer Security (ESORICS'14)*, ser. LNCS, vol. 8713. Springer, 2014, pp. 327–344.

20. P. Y. A. Ryan and S. A. Schneider, "Prêt-à-voter with re-encryption mixes." in *11th European Symp. On Research In Computer Security (ESORICS'06)*, ser. LNCS, vol. 4189. Springer, 2006, pp. 313–326.
21. J. Dreier, P. Lafourcade, and Y. Lakhnech, "Defining privacy for weighted votes, single and multi-voter coercion," in *17th European Symp. On Research In Computer Security (ESORICS'06)*, ser. LNCS, 2012.
22. A. Fujioka, T. Okamoto, and K. Ohta, "A practical secret voting scheme for large scale elections." in *Advances in Cryptology – AUSCRYPT '92*, ser. LNCS, vol. 718. Springer, 1992, pp. 244–251.
23. H. Comon-Lundh and V. Cortier, "Security properties: two agents are sufficient," *Science of Computer Programming*, vol. 50, no. 1-3, pp. 51–71, March 2004.
24. V. Cortier, A. Dallon, and S. Delaune, "Bounding the number of agents, for equivalence too," in *Proceedings of the 5th International Conference on Principles of Security and Trust (POST'16)*, ser. LNCS. Springer, 2016.
25. B. Blanchet, M. Abadi, and C. Fournet, "Automated verification of selected equivalences for security protocols," *Journal of Logic and Algebraic Programming*, vol. 75, no. 1, pp. 3–51, 2008.
26. D. Bernhard, V. Cortier, D. Galindo, O. Pereira, and B. Warinschi, "A comprehensive analysis of game-based ballot privacy definitions," in *Proceedings of the 36th IEEE Symp. on Security and Privacy (S&P'15)*. IEEE Comp. Soc., May 2015, pp. 499–516.
27. P. Bulens, D. Giry, and O. Pereira, "Running mixnet-based elections with helios," in *2011 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, EVT/WOTE '11*. USENIX Association, 2011.
28. A. Juels, D. Catalano, and M. Jakobsson, "Coercion-resistant electronic elections," in *ACM workshop on Privacy in the electronic society (WPES'05)*. ACM, 2005, pp. 61–70, see also http://www.rsa.com/rsalabs/node.asp?id=2860.
29. M. Clarkson, S. Chong, and A. Myers, "Civitas: Toward a secure voting system," in *29th IEEE Symp. on Security and Privacy (S&P'08)*. IEEE Comp. Soc., 2008, pp. 354–368.
30. V. Cheval and B. Blanchet, "Proving more observational equivalences with proverif," in *Proc. 2nd International Conference on Principles of Security and Trust (POST'13)*, ser. LNCS, vol. 7796. Springer, 2013, pp. 226–246.
31. V. Cheval, V. Cortier, and S. Delaune, "Deciding equivalence-based properties using constraint solving," *Theoretical Computer Science*, vol. 492, pp. 1–39, 2013. [Online]. Available: https://hal.inria.fr/hal-00881060

A Operational semantics

The operational semantics of our calculus is defined by a labelled transition system that allows to reason about processes which interact with their environment. The transition relation is defined by the set of labelled rules given in Figure **??**. These rules define the relation $A \xrightarrow{\ell} B$ where A and B are ground extended processes and the label ℓ is either an input ($c(M)$), an output ($\nu x.\overline{c}\langle x\rangle$), or a silent action ($\tau$). The silent actions are mostly standard, while visible input and output actions are interactions with the adversary on public channels. The label reflects the adversary's view of this interaction. The output label $\nu x.\overline{c}\langle x\rangle$ indeed reflects that messages are output "by reference": the label merely contains the variable in the domain of Φ which maps to the corresponding ground message. The input label $c(M)$ contains the term M used by the adversary to compute the message: in his computation the adversary may indeed use terms that were

$$(\mathcal{E}; \{i : \mathsf{if}\ M = N\ \mathsf{then}\ Q_1\ \mathsf{else}\ Q_2\ \} \uplus \mathcal{P}; \Phi; i) \xrightarrow{\tau} (\mathcal{E}; i : Q_1 \uplus \mathcal{P}; \Phi; i) \quad \text{if } M =_{\mathsf{E}} N \quad \text{(THEN)}$$

$$(\mathcal{E}; \{i : \mathsf{if}\ M = N\ \mathsf{then}\ Q_1\ \mathsf{else}\ Q_2\ \} \uplus \mathcal{P}; \Phi; i) \xrightarrow{\tau} (\mathcal{E}; i : Q_2 \uplus \mathcal{P}; \Phi; i) \quad \text{if } M \neq_{\mathsf{E}} N \quad \text{(ELSE)}$$

$$(\mathcal{E}; \{i : \overline{c}\langle M\rangle.Q_1; i : c(x).Q_2\} \uplus \mathcal{P}; \Phi; i) \xrightarrow{\tau} (\mathcal{E}; \{i : Q_1; i : Q_2\{x \mapsto M\}\} \uplus \mathcal{P}; \Phi; i) \quad \text{(COMM)}$$

$$(\mathcal{E}; \{i : \overline{c}\langle M\rangle.Q\} \uplus \mathcal{P}; \Phi; i) \xrightarrow{\nu y.\overline{c}\langle y\rangle} (\mathcal{E}; \{i : Q\} \uplus \mathcal{P}; \Phi \cup \{y \mapsto M\}; i) \quad \text{(OUT)}$$
$$\text{if } c \notin \mathcal{E}$$

$$(\mathcal{E}; \{i : c(x).Q\} \uplus \mathcal{P}; \Phi; i) \xrightarrow{c(M)} (\mathcal{E}; \{i : Q\{x \mapsto M\Phi\}\} \uplus \mathcal{P}; \Phi; i) \quad \text{(IN)}$$
$$\text{if } c \notin \mathcal{E},\ fv(M) \subseteq \mathrm{dom}(\Phi),\ fn(M) \cap \mathcal{E} = \emptyset, \text{ and } y \text{ is a fresh variable}$$

$$(\mathcal{E}; \{i : \nu\, n.Q\} \uplus \mathcal{P}; \Phi; i) \xrightarrow{\tau} (\mathcal{E} \cup \{m\}; \{i : Q\{n \mapsto m\}\} \uplus \mathcal{P}; \Phi; i) \quad \text{(NEW)}$$
$$\text{if } m \text{ is a fresh name}$$

$$(\mathcal{E}; \{i : !Q\} \uplus \mathcal{P}; \Phi; i) \xrightarrow{\tau} (\mathcal{E}; \{i : !Q; i : Q\} \uplus \mathcal{P}; \Phi; i) \quad \text{(REPL)}$$

$$(\mathcal{E}; \{i : (P_1 \mid P_2)\} \uplus \mathcal{P}; \Phi; i) \xrightarrow{\tau} (\mathcal{E}; \{i : P_1; i : P_2\} \uplus \mathcal{P}; \Phi; i) \quad \text{(PAR)}$$

$$(\mathcal{E}; \mathcal{P}; \Phi; i) \xrightarrow{\mathsf{phase}\ j} (\mathcal{E}; \mathcal{P}; \Phi; j) \quad \text{if } j > i \quad \text{(PHASE)}$$

$$(\mathcal{E}; \{i : j : P\} \uplus \mathcal{P}; \Phi; i) \xrightarrow{\tau} (\mathcal{E}; \{j : P\} \uplus \mathcal{P}; \Phi; i) \quad \text{(MOVE)}$$

where c is a channel, M, N, N_1, N_2 are terms, x is a variable, and i is an integer.

Fig. 3. Semantics

previously output, i.e., terms in Φ adressed through variables in $\mathrm{dom}(\Phi)$, while he is not allowed to directly use a private name.

B Some more detailed definitions

Definition 8. *A check* check *is a predicate on a sequence of terms* $V_1, \ldots, V_m$ *such that*

- *there exists* $1 \leq i_1 < i_2 \leq m$*, such that for any* $1 \leq j_1 < \cdots < j_\ell \leq m$*,*
  if $\mathsf{check}([V_1, \ldots, V_m]) = \top$ *and* $\{i_1, i_2\} \subseteq \{j_1, \ldots, j_\ell\}$
  then $\mathsf{check}([V_{j_1}, \ldots, V_{j_\ell}]) = \top$
- *for any* $V_1, \ldots, V_m$*, any* $1 \leq j_1 < \cdots < j_\ell \leq m$*,*
  if $\mathsf{check}([V_1, \ldots, V_m]) = \bot$ *then* $\mathsf{check}([V_{j_1}, \ldots, V_{j_\ell}]) = \bot$
- *for any* $V_1, \ldots, V_m$*, any* $W_1, \ldots, W_m$ *such that for all* $i \in \{1, \ldots, m\}$*,* $W_i \in \{V_i, \bot\}$
  if $\mathsf{check}([V_1, \ldots, V_m]) = \bot$ *then* $\mathsf{check}([W_1, \ldots, W_m]) = \bot$

Definition 9. *An e-voting protocol*

$$\{\Pi^{n_h, n_d, m}(\mathcal{C}r^h_{n_h}, \mathcal{C}r^d_{n_d}, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}})\}_{n_h, n_d, m \in \mathbb{N}}$$

is voting-friendly *w.r.t.* check, $\{\mathsf{Test}^m\}_{m\in\mathbb{N}}$, $\{\mathsf{Policy}^{n,m}\}_{n,m\in\mathbb{N}}$, $\mathsf{Extract}^m$, $\{\mathsf{Count}^\ell\}_{\ell\in\mathbb{N}}$ *if it is almost determinate, if* $\{\mathsf{Test}^m\}_{m\in\mathbb{N}}$, $\{\mathsf{Policy}^{n,m}\}_{n,m\in\mathbb{N}}$, $\mathsf{Extract}^m$, *are voting-friendly, and if for all*

$$(tr'\cdot \mathsf{tall} : bb(RB_1)\ldots bb(RB_m), A_1) \in \mathsf{traces}(\Pi^{n_h,n_d,m}(\mathcal{C}r^h_{n_h}, \mathcal{C}r^d_{n_d}, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}))$$

and $BB = [B_1, \ldots, B_m]$ *where* $B_i = RB_i\phi(A_1)$ *for all* $1 \le i \le m$ *we have that*

1. $(\nu y.\overline{tal}\langle y\rangle, A_2) \in \mathsf{traces}(A_1)$ *iff* $\mathsf{Test}^m(BB) = \top \wedge \mathsf{check}(BB) = \top$
2. *for all* $(\nu y.\overline{tal}\langle y\rangle, A_2) \in \mathsf{traces}(A_1)$*, we have that* $y\phi(A_2) = \langle res, nvotes, zkp\rangle$ *and*
   - $nvotes = k$, $[V_1, \ldots, V_k] = \mathsf{Policy}^{n,m}(\mathsf{Extract}^m(BB, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}), \mathcal{C}r_n)$ *where* $\mathcal{C}r_n = \mathcal{C}r^h_{n_h} \cup \mathcal{C}r^d_{n_d}$*, and* $res =^{\#} \mathsf{Count}^k([V_1, \ldots, V_k])$*;*
   - $\phi(A_2) = \nu r.\nu\tilde{n}.(\sigma \cup \{\langle res, nvotes, zkp\rangle/y\})$ *and* $zkp = \mathsf{zkp}(r, M_1, \ldots, M_\ell)$ *for some terms* $M_1, \ldots, M_\ell$ *and some fresh name* r*;*
   - $\mathsf{checkzkp}(zkp, (BB, \mathcal{K}_{\mathsf{pb}})) = \mathsf{okzkp}$ *and* $\mathsf{checkzkp}(zkp, (BB', \mathcal{K}_{\mathsf{pb}}), zkp) \neq \mathsf{okzkp}$ *for all* $BB' \neq BB$*;*
   - *either for all* $(\nu y.\overline{tal}\langle y\rangle, A'_2) \in \mathsf{traces}(A_1)$ *we have that* $y\phi(A_2) = y\phi(A'_2)$ *or for all* $res' =^{\#} res$ *there exists* $(\nu y.\overline{tal}\langle y\rangle, A'_2) \in \mathsf{traces}(A_1)$ *such that* $y\phi(A'_2) = \langle res', nvotes, zkp\rangle$.

The definition states that we only consider voting protocols that

1. output a result if and only if the public test and property are satisfied;
2. the result is a triple $\langle res, nvotes, zkp\rangle$ where res is the result computed by applying the extraction function on the bulletin board, the revote policy and the counting function; $nvotes$ is the number of votes that has been counted; and zkp is a zero knowledge proof (computed using some fresh randomness) which only holds on the current bulletin board. Moreover, for a given bulletin board either all possible traces yield the same result, or all permutations of the result are possible: if for example a trace yields the result $[V_1, V_2, V_3]$ and another trace gives $[V_1, V_3, V_2]$ then there exist traces for the results $[V_2, V_1, V_3]$, $[V_2, V_3, V_1]$, $[V_3, V_1, V_2]$ and $[V_3, V_2, V_1]$. Note that we leave the equational theory of zkp unspecified. Which proof to use depends on the protocol under consideration. Our reduction result would also for protocols without proof of correct decryption.

C Properties of e-voting protocols

Before proving our results in the next sections we state three basic properties enjoyed by voting-friendly protocols.

Our first property states that frames obtained during the voting phase are independent of the number of (dishonest) voters or even of ballots to be tallied.

Lemma 1. *Let* $\{\Pi^{n_h,n_d,m}(\mathcal{C}r^h_{n_h}, \mathcal{C}r^d_{n_d}, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}})\}_{n_h,n_d,m\in\mathbb{N}}$ *be a voting-friendly electronic voting protocol,* $\mathcal{E}_0$ *a set of names, and* Φ_0 *a ground substitution representing the initial attacker knowledge. Let* $n_h, n_d, n'_d, m, m' \in \mathbb{N}$ *such that* $n_h \ge 2$ *and let*

$$A_0 = (\mathcal{E}_0, \Pi^{n_h,n_d,m}(\mathcal{C}r^h_{n_h}, \mathcal{C}r^d_{n_d}, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}), \Phi_0, \emptyset) \quad \textit{and}$$
$$A'_0 = (\mathcal{E}_0, \Pi^{n_h,n'_d,m'}(\mathcal{C}r^h_{n_h}, \mathcal{C}r^d_{n'_d}, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}), \Phi_0, \emptyset).$$

For all traces $(tr, A) \in \mathsf{traces}(A_0)$*, if* $\mathsf{phase\ tall}$ *does not occur in* tr*, then there exists* A' *such that* $(tr, A') \in \mathsf{traces}(A'_0)$ *and* $\phi(A) = \phi(A')$*.*

Proof. The lemma follows from the semantics of our calculus and the structure of the voting protocols we consider. In particular, as tr does not contain $\mathsf{phase\ tall}$ all transitions are on the n_h honest voter processes, which coincide for all n_d, n'_d. (Recall that dishonest voters are subsumed by the environment and not modelled.)

The second property states that once the tally starts the voting process accepts to input any k votes for $k \leq m$ without modifying the frame any further, i.e. no outputs are produced while collecting the votes.

Lemma 2. *Let* $\{\Pi^{n_h,n_d,m}(\mathcal{C}r^h_{n_h}, \mathcal{C}r^d_{n_d}, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}})\}_{n_h,n_d,m\in\mathbb{N}}$ *be a voting-friendly electronic voting protocol,* $\mathcal{E}_0$ *a set of names, and* Φ_0 *an active substitution representing the initial attacker knowledge. Let* $n_h, n_d, m, \in \mathbb{N}$ *such that* $n_h \geq 2$*. Finally, let* $A_0 = (\mathcal{E}_0, \Pi^{n_h,n_d,m}(\mathcal{C}r^h_{n_h}, \mathcal{C}r^d_{n_d}, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}), \Phi_0, \emptyset)$*. For all traces* $(tr, A) \in \mathsf{traces}(A_0)$ *with* $A = (\mathcal{E}; \mathcal{P}; \Phi)$*, if*

$$tr = tr' \cdot \mathsf{phase\ tall}$$

then for all recipes $RB_1, \ldots, RB_k$ *(i.e.* $fv(RB_j) \subseteq \mathrm{dom}(\Phi)$ *and* $fn(RB_j) \cap \mathcal{E} = \emptyset$ *for all* $j \in \{1, \ldots, k\}$*) with* $1 \leq k \leq m$*, there exists* A' *such that* $(tr'', A') \in \mathsf{traces}(A'_0)$ *and* $\phi(A) = \phi(A')$ *where*

$$tr'' = tr \cdot bb(RB_1) \cdot \ldots \cdot bb(RB_k)$$

Proof. Again the lemma follows from the semantics of our calculus and the structure of the voting protocols (and in particular of the tallying process) we consider.

The final property states that adding a zero knowledge proof does not break static equivalence, unless the proof holds in one frame and not in the other.

Lemma 3. *Let* $\phi_1 = \nu\omega_1.\ \theta_1$ *and* $\phi_2 = \nu\omega_2.\ \theta_2$ *be two frames,* r *be a fresh name, and* x *a fresh variable. Let*

$$R_i = \mathsf{zkp}(r, M^i_1, \ldots, M^i_\ell)\downarrow \quad \textit{for } 1 \leq i \leq 2$$

such that for all recipes R (i.e. *for* $1 \leq j \leq 2$*,* $fv(R) \subseteq \mathrm{dom}(\phi_j)$ *and* $fn(R) \cap \omega_j = \emptyset$*),*

$$(\mathsf{checkzkp}(R_1, R) = \mathsf{okzkp})\phi_1 \textit{ iff } (\mathsf{checkzkp}(R_2, R) = \mathsf{okzkp})\phi_2.$$

We have that

$$\phi_1 \sim \phi_2 \textit{ iff } \nu\omega_1.\nu r.\ (\theta_1 \mid \{R_1/x\}) \sim \nu\omega_2.\nu r.\ (\theta_2 \mid \{R_2/x\})$$

Proof. Let $\phi'_i = \nu\omega_i.\nu r.\ (\theta_i \mid \{R_i/x\})$ for $1 \leq i \leq 2$. Note first that for $i \in \{1, 2\}$ we have $R_i = \mathsf{zkp}(r, M^i_1\downarrow, \ldots, M^i_\ell\downarrow)$.

We define $\overline{T}$ as follows:

$$\overline{T} = \begin{cases} r_0 & \text{if } T \in \{R_1, R_2\} \\ T & \text{if } T \in \mathcal{N} \cup \mathcal{X} \\ g(\overline{T_1}, \ldots, \overline{T_k}) & \text{if } T = g(T_1, \ldots, T_k) \end{cases}$$

as well as $\overline{T}^i$ for $i \in \{1, 2\}$ as follows

$$\overline{T}^i = \begin{cases} R_i & \text{if } T = r_0 \\ T & \text{if } T \in \mathcal{N} \cup \mathcal{X} \setminus \{r_0\} \\ g(\overline{T_1}^i, \ldots, \overline{T_k}^i) & \text{if } T = g(T_1, \ldots, T_k) \end{cases}$$

We prove this lemma by contradiction. Assume $\phi_1' \not\sim \phi_2'$ and let (M', N') be a minimal witness of that, that is two terms such that $(M' =_{\mathsf{E}} N')\phi_1'$ but $(M' \neq_{\mathsf{E}} N')\phi_2'$, and such that further for all term K and L if $(K =_{\mathsf{E}} L)\phi_1'$ but $(K \neq_{\mathsf{E}} L)\phi_2'$ then $|M'| + |N'| \leq |K| + |L|$. Let r_0 be a fresh name.

Claim 1. Let L be a ground term. If L is in normal form, then so is $\overline{L}$.

Claim 2. Let L be a ground term such that for any position $p \in \mathrm{pos}(L)$ if $L|_p \in \{R_1, R_2\}$ then for any position $q < p$ $L|_q \neq_{\mathsf{E}} \mathsf{okzkp}$. Let L' be a ground term such that $L \to^* L'$. Then

- for any position $p \in \mathrm{pos}(L')$ if $L'|_p \in \{R_1, R_2\}$ then for any position $q < p$ $L'|_q \neq_{\mathsf{E}} \mathsf{okzkp}$, and
- $\overline{L} \to^* \overline{L'}$.

Proof (Proof of Claim 2). Let $L = L_0 \to L_1 \ldots L - n - 1 \to L_n = L$ be a valid derivation. The proof of Claim 2 is done by induction on the size n of this derivation.

Base case ($n = 0$). In that case we have $L = L'$, and thus $\overline{L} = \overline{L'}$. So we can trivially conclude.

Inductive case ($n > 0$). In that case we know by induction hypothesis that

- for any position $p \in \mathrm{pos}(L_{n-1})$ if $L_{n-1}|_p \in \{R_1, R_2\}$ then for any position $q < p$ $L_{n-1}|_q \neq_{\mathsf{E}} \mathsf{okzkp}$, and
- $\overline{L} \to^{n-1} \overline{L_{n-1}}$.

We prove that L_n, $\overline{L_{n-1}}$ and $\overline{L_n}$ satisfy the two desired properties by induction on the size of L_{n-1}. The base case $|L_{n-1}| = 1$ cannot occur. Indeed, in that case we would have $L_{n-1} \in \mathcal{N}$ wich is in normal form and thus cannot be reduced to a different L_n. Let us now consider the inductive case $|L_{n-1}| > 1$, *i.e.* $L_{n-1} = g(W_1, \ldots, W_m)$. We distinguish several case depending on the function symbol g.

If $g = \mathsf{zkp}$. In that case $L_{n-1} \neq R_i$, and the last reduction cannot occur in head position, *i.e.* there exists $i \in \{1, \ldots, m\}$ such that $W_i \to^1 W_i'$. We can thus conclude by induction.

If $g = \mathsf{checkzkp}$. If the reduction does not occur in head position we conclude by induction. Otherwise $L_n = \mathsf{okzkp}$, but then we know that $W_1 \notin \{R_1, R_2\}$ and because by assumtion on E the zkp symbols are not inspected further inside L_{n-1}, we know that $\overline{L_n} \to \mathsf{okzkp}$. Finally $L_n = \mathsf{okzkp}$ trivially satisfies the first property.

Otherwise we distinguish two cases. If the reduction does not occur in head position we conclude by induction. If the reduction occurs in head position we know the zkp symbols are not inspected further inside L_{n-1} nor introduced in L_n, we can thus finish by induction.

Claim 3 Let L be any ground term and L' a ground term such that $L \to^* L'$. Then for $i \in \{1, 2\}$

- $\overline{L}^i \to^* \overline{L'}^i$.

Proof (Proof of Claim 3). Let $L = L_0 \to L_1 \dots L - n - 1 \to L_n = L$ be a valid derivation. The proof of Claim 2 is done by induction on the size n of this derivation.

Claim 4. For any position $p \in \mathrm{pos}(M')$, $M'|_p\phi'_1 =_{\mathsf{E}} \mathsf{okzkp}$ if and only if $M'|_p\phi'_2 =_{\mathsf{E}} \mathsf{okzkp}$. Similarily, for any position $p \in \mathrm{pos}(N')$, $N'|_p\phi'_1 =_{\mathsf{E}} \mathsf{okzkp}$ if and only if $N'|_p\phi'_2 =_{\mathsf{E}} \mathsf{okzkp}$.

Proof (Proof of Claim 1). We prove this by contradiction. Assume there exists a position $p \in \mathrm{pos}(M')$ such that $M'|_p\phi'_1 =_{\mathsf{E}} \mathsf{okzkp}$ but $M'|_p\phi'_2 \neq_{\mathsf{E}} \mathsf{okzkp}$. If $p > \epsilon$ then (M', N') is not a minimal witness of $\phi'_1 \not\sim \phi'_2$. So $p = \epsilon$ and for all positions $q \in \mathrm{pos}(M')$ with $q \neq \epsilon$, $M'|_q\phi'_1 =_{\mathsf{E}} \mathsf{okzkp}$ if and only if $M'|_q\phi'_2 =_{\mathsf{E}} \mathsf{okzkp}$. But then for (M', N') to be minimal, it must further be that for all positions $q \in \mathrm{pos}(M')$ with $q \neq \epsilon$, if $M'|_q\phi'_1 =_{\mathsf{E}} \mathsf{okzkp}$ then $M'|_q = \mathsf{okzkp}$.

If M' was a variable. Then by assumption on E, $M' \neq x$ and thus $M'\phi'_1 = M'\phi_1 =_{\mathsf{E}} \mathsf{okzkp}$ but $M'\phi'_2 = M'\phi_2 \neq_{\mathsf{E}} \mathsf{okzkp}$. However, this would be a witness of $\phi_1 \not\sim \phi_2$ with contradicts our hypothesis. Thus $M' = \mathsf{checkzkp}(U, V_1, \dots, V_m)$. We distinguish several cases.

Case $U = x$. Then for $i \in \{1, 2\}$, $U\phi'_i = R_i$ and by hypothesis we have that $\mathsf{checkzkp}(R_1, V_1\phi_1, \dots, V_m\phi_1) =_{\mathsf{E}} \mathsf{okzkp}$ if and only if $\mathsf{checkzkp}(R_2, V_1\phi_2, \dots, V_m\phi_2) =_{\mathsf{E}} \mathsf{okzkp}$. So this case cannot occur.

Case $U \neq x$. Then there exists in E a rule $\mathsf{checkzkp}(\mathsf{zkp}(U_1^{\mathsf{E}}, \dots, U_n^{\mathsf{E}}), V_1^{\mathsf{E}}, \dots, V_m^{\mathsf{E}}) \to \mathsf{okzkp}$ and a substitution σ such that $U\phi'_1 \downarrow =_{\mathrm{AC}} \mathsf{zkp}(U_1^{\mathsf{E}}\sigma, \dots, U_n^{\mathsf{E}}\sigma)$, $V_1\phi'_1 \downarrow =_{\mathrm{AC}} V_1^{\mathsf{E}}\sigma$, …, and $V_m\phi'_1 \downarrow =_{\mathrm{AC}} V_m^{\mathsf{E}}\sigma$. Now according to Claim 2 and 1 and because zkp does not occur in $U_1, \dots, U_n, V_1, \dots, V_m$ and are not AC, $\overline{U\phi'_1} \to^* \overline{U\phi'_1\downarrow} =_{\mathrm{AC}} \overline{U\phi'_1\downarrow\downarrow} =_{\mathrm{AC}} \mathsf{zkp}(U_1^{\mathsf{E}}\overline{\sigma}, \dots, U_n^{\mathsf{E}}\overline{\sigma})$, $\overline{V_1\phi'_1} \to^* \overline{V_1\phi'_1\downarrow} =_{\mathrm{AC}} \overline{V_1\phi'_1\downarrow\downarrow} =_{\mathrm{AC}} V_1^{\mathsf{E}}\overline{\sigma}$, …, $\overline{V_m\phi'_1} \to^* \overline{V_m\phi'_1\downarrow} =_{\mathrm{AC}} \overline{V_m\phi'_1\downarrow\downarrow} =_{\mathrm{AC}} V_m^{\mathsf{E}}\overline{\sigma}$. But then we have that $M\phi_1 = \overline{M'\phi'_1} \to^* \mathsf{checkzkp}(\overline{U\phi'_1}, V_1\phi'_1, \dots, V_m\phi'_1) \to \mathsf{okzkp}$. Now since $\phi_1 \sim \phi_2$, it must also be the case that $M\phi_2 \to^* \mathsf{okzkp}$, from which we can trivially conclude that $M'\phi'_2 \to^* \mathsf{okzkp}$ which contradicts our hypothesis.

Note that by Claim 4 and by minimality of (M', N'), it must be that for any $i \in \{1, 2\}$ and any position p in M' (*resp.* in N') if $M'\phi'_i =_{\mathsf{E}} \mathsf{okzkp}$ (*resp.* $N'\phi'_i =_{\mathsf{E}} \mathsf{okzkp}$) then $M' = \mathsf{okzkp}$ (*resp.* $N' = \mathsf{okzkp}$). In particular this means that for any $i \in \{1, 2\}$ and any position p in $M'\phi'_i$ (*resp.* $N'\phi'_i$) if $M'\phi'_i|_p = R_i$ (*resp.* $M'\phi'_i|_p = R_i$) then for any position $q < p$ $M'\phi'_i|_q \neq_{\mathsf{E}} \mathsf{okzkp}$ (*resp.* $N'\phi'_i|_q \neq_{\mathsf{E}} \mathsf{okzkp}$).

We are now going to show that for $M = M'\{r_0/x\}$ and $N = N'\{r_0/x\}$ we have $(M =_{\mathsf{E}} N)\phi_1$ but $(M \neq_{\mathsf{E}} N)\phi_2$. More precisely what we are going to show is that for $i \in \{1, 2\}$, $(M' =_{\mathsf{E}} N')\phi'_i$ if and only if $(M =_{\mathsf{E}} N)\phi_i$.

$(M' =_{\mathsf{E}} N')\phi'_i \Rightarrow (M =_{\mathsf{E}} N)\phi_i$.

According to this Claim 2 we have $M\phi_i = \overline{M'\phi'_i} \to^* \overline{M'\phi'_i\downarrow}$ and $N\phi_i = \overline{N'\phi'_i} \to^* \overline{N'\phi'_i\downarrow}$. Now by hypothesis we have that $M'\phi'_i\downarrow =_{\mathrm{AC}} N'\phi'_i\downarrow$, and thus that $\overline{M'\phi'_i\downarrow} =_{\mathrm{AC}}$

$\overline{N'\phi'_i\downarrow}$ because checkzkp and zkp are not AC. Moreover by Claim 1, we also have that $\overline{M'\phi'_i\downarrow}$ and $\overline{N'\phi'_i\downarrow}$ are in normal form. So we can conclude that $\overline{M'\phi'_i\downarrow} = M\phi_i\downarrow$ and $\overline{N'\phi'_i\downarrow} = N\phi_i\downarrow$. Combining all these we derive that $M\phi_i\downarrow =_{\mathrm{AC}} N\phi_i\downarrow$ and thus that $(M =_{\mathsf{E}} N)\phi$.

$(M =_{\mathsf{E}} N)\phi_i \Rightarrow (M' =_{\mathsf{E}} N')\phi'_i$.

Assume $(M =_{\mathsf{E}} N)\phi_i$, this by definition means that $M\phi_i\downarrow =_{\mathrm{AC}} N\phi_i\downarrow$. Now by Claim 3 we have that $M'\phi'_i = \overline{M\phi_i}^i \rightarrow \overline{M\phi_i\downarrow}^i$ and $N'\phi'_i = \overline{N\phi_i}^i \rightarrow \overline{N\phi_i\downarrow}^i$. Thus, because our rewritte theory is AC-convergent, $M'\phi'_i\downarrow =_{\mathrm{AC}} \overline{M\phi_i\downarrow}^i\downarrow =_{\mathrm{AC}} \overline{N\phi_i\downarrow}^i\downarrow =_{\mathrm{AC}} M'\phi'_i\downarrow$ which by definition means that $(M' =_{\mathsf{E}} N')\phi'_i$.

D Bounding the number of voters (proof of Proposition 1)

This section is dedicated to the proof that if there is an attack involving n voters, then there is an attack involving at most 3 voters.

We first start by establishing that trace equivalence is closed under application of evaluation contexts.

Definition 10. *An evaluation context $(\mathcal{E} \cup \_; \mathcal{P} \cup \_; \Phi \cup \_; \_)$ is an extended process with holes "_" for bound names, plain processes, ground substitution, and phase. Let $C[\_] = (\mathcal{E} \cup \_; \mathcal{P} \cup \_; \Phi \cup \_; \_)$ be an evaluation context and $A = (\mathcal{E}_A; \mathcal{P}_A; \Phi_A; i_A)$ be a closed extended process with $\mathcal{E}_A \cap (\mathcal{E} \cup fn(\Phi) \cup fn(\mathcal{P})) = \mathrm{dom}(\Phi_A) \cap \mathrm{dom}(\Phi) = \emptyset$. Applying $C[\_]$ to A gives the extended process:*

$$C[A] = (\mathcal{E} \cup \mathcal{E}_A; \mathcal{P} \cup \mathcal{P}_A; \Phi \cup \Phi_A; i_A)$$

An evaluation context $C[\_]$ closes A when $C[A]$ is a closed extended process.

Lemma 4. *Let $A = (\mathcal{E}_A; \mathcal{P}_A; \Phi_A; i)$ and $B = (\mathcal{E}_B; \mathcal{P}_B; \Phi_B; i)$ be two closed extended processes, and let $C[\_]$ be a closing evaluation context for A and B. If $A \approx B$, then $C[A] \approx C[B]$.*

Proof. The proof is similar to the proof of Proposition 1 in [31].

The following lemma allows to reduce from k_h honest voters and k_d dishonest ones, to only 2 honest voters and the other $k_h + k_d - 2$ voters being dishonest.

Lemma 5. *If $A_0^{k_h,k_d,\ell} \not\approx B_0^{k_h,k_d,\ell}$ then $A_0^{2,k_h+k_d-2,\ell} \not\approx B_0^{2,k_h+k_d-2,\ell}$.*

Proof. This is because trace equivalence is closed under application of evaluation contexts (Lemma **??**).

Proof (of Proposition 1). Suppose there exists a trace $(tr_C, C) \in \mathsf{traces}(A_0^{n_h,n'_d,m})$, such that for any process D, if $(tr_C, D) \in \mathsf{traces}(B_0^{n_h,n'_d,m})$, then $C \not\sim D$, *i.e.* (tr_C, C) is an attack involving n_h honest voters, n'_d dishonest voters, and m ballots. Let $n_d = n_h + n'_d - 2$. By Lemma **??** we know that we have an attack that involves the same

number of voters but only two of them being honest. In other words, there exists a trace $(tr, A) \in \mathsf{traces}(A_0^{2,n_d,m})$, such that for any process B, if $(tr, B) \in \mathsf{traces}(B_0^{2,n_d,m})$, then $A \not\sim B$, *i.e.* (tr, A) is an attack involving 2 honest voters, $n_d = n_h + n'_d - 2$ dishonest voters, and m ballots.

Let us consider (tr, A) to be minimal *w.r.t.* the length of tr. According to the structure of a voting protocol all traces are of the form

$$tr = tr' \cdot \mathsf{phase\ tall} \cdot bb(RB_1) \ldots bb(RB_m) \cdot \nu y.\overline{tal}\langle y \rangle$$

and we distinguish four cases:

1. $\mathsf{phase\ tall}$ does not occur in tr, *i.e.* the attack has occured during the vote casting phase. By Lemma **??**, there exists A' such that $(tr, A') \in \mathsf{traces}(A_0^{2,0,m})$ and $\phi(A) = \phi(A')$. If (tr, A') is not an attack, then there exists B' such that $(tr, B') \in \mathsf{traces}(B_0^{2,0,m})$ and $\phi(A') \sim \phi(B')$. But then, by Lemma **??** again there should exist B such that $(tr, B) \in \mathsf{traces}(B_0^{2,n_d,m})$ and $\phi(B') = \phi(B)$. Therefore B would be such that $\phi(A) = \phi(A') \sim \phi(B') = \phi(B)$ which contradicts our hypothesis that (tr, A) is an attack. Hence, we can conclude that (tr, A') is an attack involving only 2 voters (both honest).

2. $tr = tr' \cdot \mathsf{phase\ tall}$ for some sequence of labels tr'. This case cannot occur. Indeed, let A' be the process such that $(tr', A') \in \mathsf{traces}(A_0^{2,n_d,m})$, $(\mathsf{phase\ tall}, A) \in \mathsf{traces}(A')$, and $\phi(A) = \phi(A')$. By minimality of tr we know that (tr', A') is not an attack, *i.e.* there exists B' such that $(tr', B') \in \mathsf{traces}(B_0^{2,n_d,m})$ and $\phi(A') \sim \phi(B')$. But by the structure of an e-voting protocol (Section 3) the phase change $\mathsf{phase\ tall}$ can be executed at any time. Hence, there exists B such that $(tr, B) \in \mathsf{traces}(B_0^{2,n_d,m})$, $\phi(B') = \phi(B)$. For (tr, A) to be an attack, it must be that $A \not\sim B$. This contradicts the hypothesis that (tr, A) is minimal as it would mean that (tr', A') is a "smaller" attack.

3. $tr = tr' \cdot \mathsf{phase\ tall} \cdot bb(RB_1) \cdot \ldots \cdot bb(RB_k)$ for some sequence of labels tr', some $1 \leq k \leq m$, and some recipes $RB_1, \ldots, RB_k$, *i.e.* the attack has occured during the tallying phase but before the output of the final tally. This case cannot occur either. Indeed, let A' be the process such that $(tr' \cdot \mathsf{phase\ tall}, A') \in \mathsf{traces}(A_0^{2,n_d,m})$, $(bb(RB_1) \cdot \ldots \cdot bb(RB_k), A) \in \mathsf{traces}(A')$, and $\phi(A) = \phi(A')$. By minimality of tr we know that $(tr' \cdot \mathsf{phase\ tall}, A')$ is not an attack, *i.e.* there exists B' such that

$$(tr' \cdot \mathsf{phase\ tall}, B') \in \mathsf{traces}(B_0^{2,n_d,m})$$

and $\phi(A') \sim \phi(B')$. As, according to our semantics, all bound names are chosen freshly we assume without loss of generality that $bn(B') \cap fn(RB_1, \ldots, RB_k) = \emptyset$. But then, by Lemma **??** there exists B such that $(tr, B) \in \mathsf{traces}(B_0^{2,n_d,m})$ and $\phi(B) = \phi(B')$. Therefore, we have that B is such that $\phi(A) = \phi(A') \sim \phi(B') = \phi(B)$ which contradicts our hypothesis that (tr, A) is an attack. Hence, we can conclude that this case cannot occur.

4. $tr = tr' \cdot \mathsf{phase\ tall} \cdot bb(RB_1) \ldots bb(RB_m) \cdot \nu y.\overline{tal}\langle y \rangle$ for some sequence of labels tr', some variable y and some recipes $RB_1, \ldots, RB_m$, *i.e.* the attack has occured after the output of the final tally.

Let $\widehat{tr} = tr'\cdot\mathsf{phase\ tall}\cdot bb(RB_1)\ldots bb(RB_m)$. And let A' be such that $(\widehat{tr}, A') \in \mathsf{traces}(A_0^{2,n_d,m})$, $(\nu y.\overline{tal}\langle y\rangle, A) \in \mathsf{traces}(A')$, and $\phi(A) = \nu\tilde{r}.\nu\tilde{n}.(\sigma \cup \{y\phi(A)/y\})$ with $\phi(A') = \nu\tilde{n}.\sigma$. Because Π_1 complies with $\{\mathsf{Test}^m\}_{m\in\mathbb{N}}$ and check (Definition 6), it must be that $\mathsf{Test}^m(BB_1) = \top$ and $\mathsf{check}(BB_1) = \top$ for $BB_1 = [B_1^1; \ldots; B_m^1]$ and $B_i^1 = RB_i\phi(A')$ for all $1 \leq i \leq m$. We distinguish two cases.

4.1. For all B, $(tr, B) \notin \mathsf{traces}(B_0^{2,n_d,m})$. Because Π_2 complies with $\{\mathsf{Test}^m\}_{m\in\mathbb{N}}$ and check (Definition 6), it must be that for all B' such that $(\widehat{tr}, B') \in \mathsf{traces}(B_0^{2,n_d,m})$ either $\mathsf{Test}^m(BB_2) = \bot$ or $\mathsf{check}(BB_2) = \bot$ for $BB_2 = [B_1^2; \ldots; B_m^2]$ and $B_i^2 = RB_i\phi(B')$ for all $1 \leq i \leq m$. But knowing Test^m is public we can conclude that $\mathsf{Test}^m(BB_2) = \bot$ would contradict the minimality of tr. Thus it must be the case that $\mathsf{Test}^m(BB_2) = \top$ and $\mathsf{check}(BB_2) = \bot$.

Let $1 \leq w_1 < w_2 \leq m$ witness that BB_1 satisfies check, that is such that $\mathsf{check}([B_{w_1}^1; B_{w_2}^1]) = \top$. By the semantics of our calculus, as well as Lemmas **??** and **??**, there must exist A'' such that $(\widehat{tr}, A'') \in \mathsf{traces}(A_0^{2,m})$ and $\phi(A') = \phi(A'')$.

Assume there exists a process B'' such that $(\widehat{tr}, B'') \in \mathsf{traces}(B_0^{2,0,m})$ and with $\mathsf{check}([RB_{w_1}\phi(B''); RB_{w_2}\phi(B'')]) = \top$. Then by the semantics of our calculus, and Lemmas **??** and **??**, there exists B' such that $(\widehat{tr}, B') \in \mathsf{traces}(B_0^{2,n_d,m})$ and $\phi(B') = \phi(B'')$. Then this would mean that $\mathsf{check}([RB_{w_1}\phi(B'); RB_{w_2}\phi(B')])$, and by the second item of the definition of a property (Definition **??**) that $\mathsf{check}(BB_2) = \top$ for $BB_2 = [B_1^2; \ldots; B_m^2]$ and $B_i^2 = RB_i\phi(B')$ for all $1 \leq i \leq m$. But we have already established that no such B' exists.

We have hence proved that $(tr, A'') \in \mathsf{traces}(A_0^{2,0,m})$ is an attack that involves only the 2 honest voters (and m ballots tallied).

4.2. For some B, $(tr, B) \in \mathsf{traces}(B_0^{2,n_d,m})$. Let $BB_1 = [B_1^1; \ldots; B_m^1]$ with $B_i^1 = RB_i\phi(A')$. Then $y\phi(A) = \langle res_1, nv_1, zkp_1\rangle$ for some res_1, nv_1, zkp_1 (Definition 6) such that

$$\underset{\mathbf{BB_1}}{\begin{bmatrix} B_1^1 \\ \cdots \\ B_m^1 \end{bmatrix}} \xrightarrow{\mathsf{Extract}_1^m(BB_1,\mathcal{K}_{\mathsf{pv}},\mathcal{K}_{\mathsf{pb}})} \underset{\mathbf{VV_1}}{\begin{bmatrix} V_1^1 \\ \cdots \\ V_m^1 \end{bmatrix}} \xrightarrow{\mathsf{Policy}_1^{n,m}(VV_1,\mathcal{C}r_n)} \underset{\mathbf{WW_1}}{\begin{bmatrix} V_{i_1^1}^1 \\ \cdots \\ V_{i_{h_1}^1}^1 \end{bmatrix}} \rightarrow \begin{array}{l}\langle res_1, nv_1, zkp_1\rangle \\ \text{s.t. } res_1 =^{\#} \mathsf{Count}^{h_1}(WW_1) \\ \text{and } nv_1 = h_1\end{array}$$

Similarily, let $BB_2 = [B_1^2; \ldots; B_m^2]$ with $B_i^2 = RB_i\phi(B')$. Then $y\phi(B) = \langle res_2, nv_2, zkp_2\rangle$ for some res_2, nv_2, zkp_2 (Definition 6) such that

$$\underset{\mathbf{BB_2}}{\begin{bmatrix} B_1^2 \\ \cdots \\ B_m^2 \end{bmatrix}} \xrightarrow{\mathsf{Extract}_2^m(BB_2,\mathcal{K}_{\mathsf{pv}},\mathcal{K}_{\mathsf{pb}})} \underset{\mathbf{VV_2}}{\begin{bmatrix} V_1^2 \\ \cdots \\ V_m^2 \end{bmatrix}} \xrightarrow{\mathsf{Policy}_2^{n,m}(VV_2,\mathcal{C}r_n)} \underset{\mathbf{WW_2}}{\begin{bmatrix} V_{i_1^2}^2 \\ \cdots \\ V_{i_{h_2}^2}^2 \end{bmatrix}} \rightarrow \begin{array}{l}\langle res_2, nv_2, zkp_2\rangle \\ \text{s.t. } res_2 =^{\#} \mathsf{Count}^{h_2}(WW_2) \\ \text{and } nv_2 = h_2\end{array}$$

According to Lemma **??** it either the case that $nv_1 \neq_{\mathsf{E}} nv_2$ or that $res_1 \neq_{\mathsf{E}} res_2$.

By Lemmas **??** and **??**, we know that for some C'

$$(tr'\cdot\mathsf{phase\ tall}\cdot bb(RB_1)\ldots bb(RB_m), C') \in \mathsf{traces}(A_0^{2,0,m})$$

with $\phi(C') = \phi(A')$. Thus $RB_i\phi(C') = RB_i\phi(A')$ for all $1 \leq i \leq m$. Also for some D'

$$(tr' \cdot \mathsf{phase\ tall} \cdot bb(RB_1) \ldots bb(RB_m), D') \in \mathsf{traces}(B_0^{2,0,m})$$

with $\phi(D') = \phi(B')$. Thus $RB_i\phi(D') = RB_i\phi(B')$ for all $1 \leq i \leq m$. We further know that $\mathsf{Test}_1^m(BB_1) = \mathsf{Test}_1^m(BB_2) = \top$ and $\mathsf{check}_1(BB_1) = \mathsf{check}_2(BB_2) = \top$. Because Π_1 and Π_2 comply with $\{\mathsf{Test}_m\}_{m\in\mathbb{N}}$ and check (Definition 6) we know that for some C

$$(tr, C) \in \mathsf{traces}(A_0^{2,0,m})$$

with $y\phi(C) = \langle res_1', nv_1', zkp_1' \rangle$, and that for some D

$$(tr, D) \in \mathsf{traces}(B_0^{2,0,m})$$

with $y\phi(D) = \langle res_2', nv_2', zkp_2' \rangle$. If for all such D, either $res_1' \neq_{\mathsf{E}} res_2'$ or $nv_1' \neq_{\mathsf{E}} nv_2'$, then $(tr, C) \in \mathsf{traces}(A_0^{2,m})$ is an attack involving only the 2 voters.

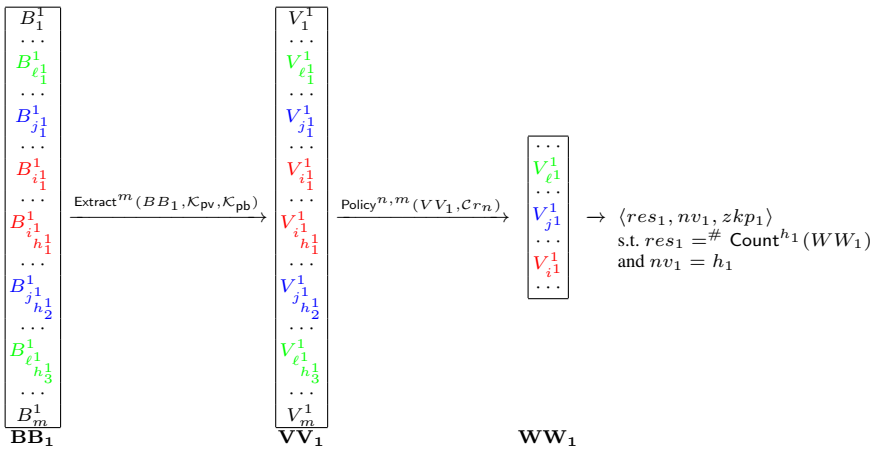
Otherwise, there exists such a D with $res_1' =_{\mathsf{E}} res_2'$ and $nv_1' =_{\mathsf{E}} nv_2'$. The first and last policies are such that $\mathsf{Policy}_1^{2,m}(VV_1) = [V_1, V_{nv_1'}]$ and $\mathsf{Policy}_2^{2,m}(VV_2) = [V_1', V_{nv_2'}']$ with $V_1, V_{nv_1'}, V_1', V_{nv_2'}' \in \{(c, \tilde{cr}) \mid c \in \mathcal{C} \text{ and } \tilde{cr} \in \{c\tilde{r}_1, c\tilde{r}_2\}\}$. Furthermore $\mathsf{Count}_1^{nv_1'} =^{\#} \mathsf{Count}_2^{nv_2'}$. We now show that in that case there must exist an attack involving only 3 voters (the two honest and one dishonest). We here assume that $nv_1' = nv_2' = 2$ but the cases $nv_1' = nv_2' = 1$ and $nv_1' = nv_2' = 0$ can be handled in a similar way.

Indeed, it must be that there exists $3 \leq i_{\mathsf{att}} \leq n$ such that for some $c \in \mathcal{C}$, it is the case that $(c, \widetilde{cr_{i_{\mathsf{att}}}}) \in WW_1$ but $(c, \widetilde{cr_{i_{\mathsf{att}}}}) \notin WW_2$ (or that $(c, \widetilde{cr_{i_{\mathsf{att}}}}) \in WW_2$ but $(c, \widetilde{cr_{i_{\mathsf{att}}}}) \notin WW_1$). Otherwise, because $\{\mathsf{Count}_i^m\}_{m\in\mathbb{N}}$ are voting friendly it couldn't be that $(tr, A) \in \mathsf{traces}(A_0^{2,n_d,m})$ is an attack.

Let $1 \leq i_1^1 < \cdots < i_{h_1^1}^1 \leq m$, be the ballots considered as casted by $\widetilde{cr_1}$, *i.e.* $\{i_1^1, \ldots, i_{h_1^1}^1\} = \{1 \leq i \leq m \mid \mathsf{Extract}^1(B_i^1, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}) = (c_i, \widetilde{cr_1}) \text{ for some } c_i \in \mathcal{C}\}$. And let i^1 be $c\tilde{r}_1$'s ballot selected by the policy.

Let $1 \leq j_1^1 < \cdots < j_{h_2^1}^1 \leq m$, be the ballots considered as casted by $\widetilde{cr_2}$, *i.e.* $\{j_1^1, \ldots, j_{h_2^1}^1\} = \{1 \leq i \leq m \mid \mathsf{Extract}^1(B_i^1, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}) = (c_i, \widetilde{cr_2}) \text{ for some } c_i \in \mathcal{C}\}$. And let j^1 be $c\tilde{r}_2$'s ballot selected by the policy.

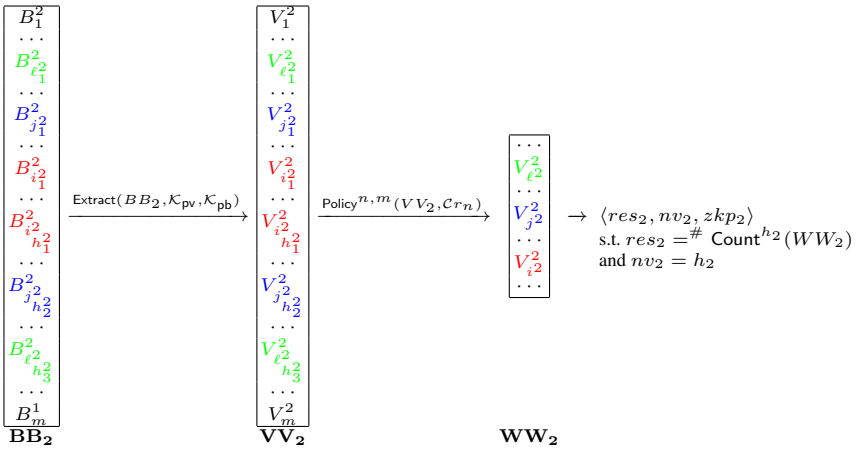
Let $1 \leq \ell_1^1 < \cdots < \ell_{h_3^1}^1 \leq m$, be the ballots considered as casted by $\widetilde{cr_{i_{\mathsf{att}}}}$, *i.e.* $\{\ell_1^1, \ldots, \ell_{h_2^1}^1\} = \{1 \leq i \leq m \mid \mathsf{Extract}^1(B_i^1, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}) = (c_i, \widetilde{cr_{i_{\mathsf{att}}}}) \text{ for some } c_i \in \mathcal{C}\}$. And let ℓ^1 be $\tilde{cr_{i_{\mathsf{att}}}}$'s ballot selected by the policy.



Let $1 \leq i_1^2 < \cdots < i_{h_1^2}^2 \leq m$, be the ballots considered as casted by $\widetilde{cr_1}$, *i.e.* $\{i_1^2, \ldots, i_{h_1^2}^2\} = \{1 \leq i \leq m \mid \mathsf{Extract}^1(B_i^2, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}) = (c_i, \widetilde{cr_1}) \text{ for some } c_i \in \mathcal{C}\}$. And let i^2 be $\tilde{cr}_1$'s ballot selected by the policy.

Let $1 \leq j_1^2 < \cdots < j_{h_2^2}^2 \leq m$, be the ballots considered as casted by $\widetilde{cr_2}$, *i.e.* $\{j_1^2, \ldots, j_{h_2^2}^2\} = \{1 \leq i \leq m \mid \mathsf{Extract}^1(B_i^2, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}) = (c_i, \widetilde{cr_2}) \text{ for some } c_i \in \mathcal{C}\}$. And let j^2 be $\tilde{cr}_2$'s ballot selected by the policy.

Let $1 \leq \ell_1^2 < \cdots < \ell_{h_3^2}^2 \leq m$, be the ballots considered as casted by $\widetilde{cr_{i_{\mathsf{att}}}}$, *i.e.* $\{\ell_1^2, \ldots, \ell_{h_3^2}^2\} = \{1 \leq i \leq m \mid \mathsf{Extract}^1(B_i^2, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}) = (c_i, \widetilde{cr_{i_{\mathsf{att}}}}) \text{ for some } c_i \in \mathcal{C}\}$. And let ℓ^2 be $\tilde{cr}_{i_{\mathsf{att}}}$'s ballot selected by the policy.



By Lemmas **??** and **??**, we know that for some C'

$$(tr' \cdot \mathsf{phase\ tall} \cdot bb(RB_1) \ldots bb(RB_m), C') \in \mathsf{traces}(A_0^{2,1,m}) \quad // \, \mathcal{C}r_3 = \{\tilde{cr}_1, \tilde{cr}_2, \tilde{cr}_{i_{\mathsf{att}}}\}$$

with $RB_i\phi(C) = RB_i\phi(A)$ for all $1 \leq i \leq m$, and that for some D'

$$(tr' \cdot \mathsf{phase}\ \mathsf{tall} \cdot bb(RB_1) \ldots bb(RB_m), D') \in \mathsf{traces}(B_0^{2,1,m}) \quad // \, \mathcal{C}r_3 = \{c\tilde{r}_1, c\tilde{r}_2, c\tilde{r}_{i_{\mathsf{att}}}\}$$

with $RB_i\phi(D) = RB_i\phi(B)$ for all $1 \leq i \leq m$. We further know that $\mathsf{Test}_1^m(BB_1) = \mathsf{Test}_1^m(BB_2) = \top$ and $\mathsf{check}_1(BB_1) = \mathsf{check}_2(BB_2) = \top$. Thus, because Π_1 and Π_2 comply with $\{\mathsf{Test}_m\}_{m \in \mathbb{N}}$ and check, we know that for some C

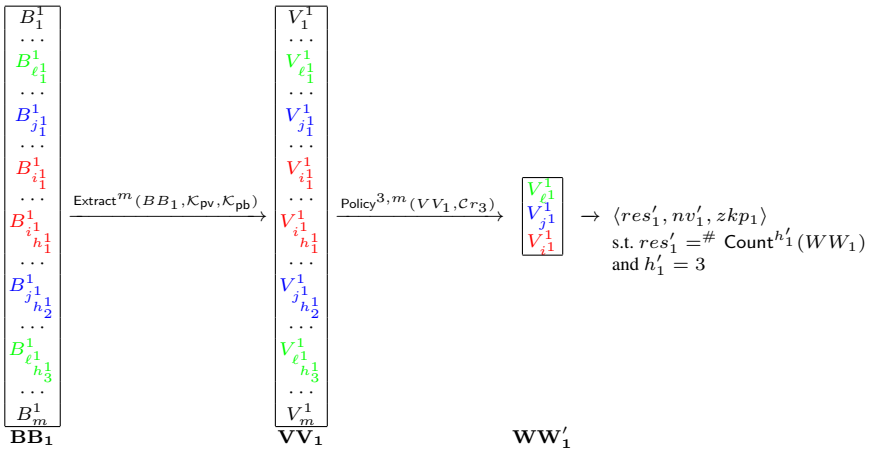
$$(tr, C) \in \mathsf{traces}(A_0^{2,1,m}) \quad // \, \mathcal{C}r_3 = \{c\tilde{r}_1, c\tilde{r}_2, c\tilde{r}_{i_{\mathsf{att}}}\}$$

with $y\phi(C) = \langle res_1', nv_1', zkp_1' \rangle$, and that for some D

$$(tr, D) \in \mathsf{traces}(B_0^{2,1,m}) \quad // \, \mathcal{C}r_3 = \{c\tilde{r}_1, c\tilde{r}_2, c\tilde{r}_{i_{\mathsf{att}}}\}$$

with $y\phi(D) = \langle res_2', nv_2', zkp_2' \rangle$.

Furthermore, by Definition 6 we know that



We distinguish two cases according to if voter $cr\tilde{}_{i_{\mathsf{att}}}$ voted in WW_2 or not.

4.2.1. There exists $c' \in \mathcal{C}$ such that $(c', cr\tilde{}_{i_{\mathsf{att}}}) \in WW_2$. In that case, $V^1_{\ell^1} = (c, cr\tilde{}_{i_{\mathsf{att}}})$ and $V^2_{\ell^2} = (c', cr\tilde{}_{i_{\mathsf{att}}})$ with $c \neq c'$. But then because the counting functions are voting friendly (Definition 4), we know that there is no permutation of res'_2 equal to res'_1

Finally, by determinacy of e-voting protocols, we know that for all E such that $(tr, E) \in \mathsf{traces}(B^{2,n_d,h})$, it must be that $y\phi(E) = \langle res''_2, nv'_2, zkp''_2\rangle$ and that res''_2 is a permutation of res'_2. We can thus conclude that for all E such that $(tr, E) \in \mathsf{traces}(B_0^{2,n_d,h})$, $C \not\sim E$ which ends the proof that $(tr, C) \in A_0^{2,1,m}$ is an attack involving 3 voters (2 honests and 1 dishonest) and m ballots.

4.2.2. There exists no $c' \in \mathcal{C}$ such that $(c', cr\tilde{}_{i_{\mathsf{att}}}) \in WW_2$. In that case $V^1_{\ell^1} = (c, cr\tilde{}_{i_{\mathsf{att}}})$, and $h'_2 = 2$. But then by determinacy of e-voting protocols, we know that for all E such that $(tr, E) \in \mathsf{traces}(B^{2,n_d,h})$, it must be that $y\phi(E) = \langle res''_2, nv'_2, zkp''_2\rangle$. We can thus conclude that for all E such that $(tr, E) \in \mathsf{traces}(B_0^{2,n_d,h})$, $C \not\sim E$ which ends the proof that $(tr, C) \in A_0^{2,1,m}$ is an attack involving 3 voters (2 honests and 1 dishonest) and m ballots.

E Bounding the number of ballots (proof of Proposition 2)

We prove here that if there is an attack involving n voters, then there is an attack involving n voters and at most $4 + 2n$ ballots tallied.

Proof (of Proposition 2). Suppose there exists a trace $(tr, A) \in \mathsf{traces}(A_0^{n_h,n_d,m})$, such that for any process B, if $(tr, B) \in \mathsf{traces}(B_0^{n_h,n_d,m})$, then $A \not\sim B$, *i.e.* (tr, A) is an attack involving $n = n_h + n_d$ voters and m ballots. Let us consider (tr, A) to be minimal *w.r.t.* the length of tr. According to the structure of a voting protocol all traces are of the form

$$tr = tr' \cdot \mathsf{phase\ tall} \cdot bb(RB_1) \ldots bb(RB_m) \cdot \nu y.\overline{tal}\langle y\rangle$$

and we distinguish four cases:

1. $\mathsf{phase\ tall}$ does not occur in tr, *i.e.* the attack has occured during the votes casting phase. Let $m_{min} = 4+2n$. By Lemma **??**, we know there exists A' such that $(tr, A') \in \mathsf{traces}(A_0^{n_h,n_d,m_{min}})$ and $\phi(A) = \phi(A')$. If (tr, A') is not an attack, then there exists B' such that $(tr, B') \in \mathsf{traces}(B_0^{n_h,n_d,m_{min}})$ and $\phi(A') \sim \phi(B')$. But then according to Lemma **??** again there should exist B such that $(tr, B) \in \mathsf{traces}(B_0^{n_h,n_d,m})$ and $\phi(B') = \phi(B)$. Combining all these we have that B would be such that $\phi(A) = \phi(A') \sim \phi(B') = \phi(B)$ which contradicts our hypothesis that (tr, A) is an attack. Hence, we can conclude that (tr, A') is an attack involving only $4 + 2n$ ballots.

2. $tr = tr' \cdot \mathsf{phase\ tall}$ for some sequence of labels tr'. This case cannot occur. Indeed, let A' be the process such that $(tr', A') \in \mathsf{traces}(A_0^{n_h,n_d,m})$, $(\mathsf{phase\ tall}, A) \in \mathsf{traces}(A')$, and $\phi(A) = \phi(A')$. By minimality of tr we know that (tr', A') is not an attack, *i.e.* there exists B' such that $(tr', B') \in \mathsf{traces}(B_0^{n_h,n_d,m})$ and $\phi(A') \sim \phi(B')$. But by the structure of an e-voting protocol (Section 3) the phase change $\mathsf{phase\ tall}$ can be executed at any time. So there exists B such that $(tr, B) \in \mathsf{traces}(B_0^{n_h,n_d,m})$, $\phi(B') = \phi(B)$. This

contradicts the hypothesis that (tr, A) is minimal as it would mean that (tr', A') is a "smaller" attack.

3. $tr = tr'\cdot\mathsf{phase\ tall}\cdot bb(RB_1)\cdot\ldots\cdot bb(RB_k)$ for some sequence of labels tr', some $1 \leq k \leq m$, and some recipes $RB_1, \ldots, RB_k$, *i.e.* the attack has occured during the tallying phase but before the output of the final tally. This case cannot occur either. Indeed, let A' be the process such that $(tr'\cdot\mathsf{phase\ tall}, A') \in \mathsf{traces}(A_0^{n_h,n_d,m})$, and such that $(bb(RB_1)\cdot\ldots\cdot bb(RB_k), A) \in \mathsf{traces}(A')$, and $\phi(A) = \phi(A')$. By minimality of tr we know that $(tr'\cdot\mathsf{phase\ tall}, A')$ is not an attack, *i.e.* there exists B' such that

$$(tr'\cdot\mathsf{phase\ tall}, B') \in \mathsf{traces}(B_0^{n_h,n_d,m})$$

and $\phi(A') \sim \phi(B')$. As, according to our semantics, all bound names are chosen freshly we assume without loss of generality that $bn(B') \cap fn(RB_1, \ldots, RB_k) = \emptyset$. But then, according to Lemma **??** there should exist B such that $(tr, B) \in \mathsf{traces}(B_0^{n_h,n_d,m})$ and $\phi(B) = \phi(B')$. Combining all these we have that B is such that $\phi(A) = \phi(A') \sim \phi(B') = \phi(B)$ which contradicts our hypothesis that (tr, A) is an attack. Hence, we can conclude that this case cannot occur.

4. $tr = tr'\cdot\mathsf{phase\ tall}\cdot bb(RB_1)\ldots bb(RB_m)\cdot\nu y.\overline{tal}\langle y\rangle$ for some sequence of labels tr', some variable y and some recipes $RB_1, \ldots, RB_m$, *i.e.* the attack has occured after the output of the final tally.

Let $\widehat{tr} = tr'\cdot\mathsf{phase\ tall}\cdot bb(RB_1)\ldots bb(RB_m)$. And let A' be such that $(\widehat{tr}, A') \in \mathsf{traces}(A_0^{n_h,n_d,m})$, $(\nu y.\overline{tal}\langle y\rangle, A) \in \mathsf{traces}(A')$, and $\phi(A) = \nu\tilde{r}.\nu\tilde{n}.(\sigma \cup \{y/\phi(A)\})$ with $\phi(A') = \nu\tilde{n}.\sigma$. Because Π_1 complies with $\{\mathsf{Test}^m\}_{m\in\mathbb{N}}$ and check (Definition 6), it must be that $\mathsf{Test}^m(BB_1) = \top$ and $\mathsf{check}(BB_1) = \top$ for $BB_1 = [B_1^1; \ldots; B_m^1]$ and $B_i^1 = RB_i\phi(A')$ for all $1 \leq i \leq m$. We distinguish two cases.

4.1. For all B, $(tr, B) \notin \mathsf{traces}(B_0^{n_h,n_d,m})$. Because Π_2 complies with $\{\mathsf{Test}^m\}_{m\in\mathbb{N}}$ and check (Definition 6), it must be that for all B' such that $(\widehat{tr}, B') \in \mathsf{traces}(B_0^{n_h,n_d,m})$ either $\mathsf{Test}^m(BB_2) = \bot$ or $\mathsf{check}(BB_2) = \bot$ for $BB_2 = [B_1^2; \ldots; B_m^2]$ and $B_i^2 = RB_i\phi(B')$ for all $1 \leq i \leq m$. But knowing Test^m is public we can conclude that $\mathsf{Test}^m(BB_2) = \bot$ would contradict the minimality of tr. Thus it must be the case that $\mathsf{Test}^m(BB_2) = \top$ and $\mathsf{check}(BB_2) = \bot$.

Let $1 \leq w_1 < w_2 \leq m$ witness that BB_1 satisfies check, that is such that $\mathsf{check}([B_{w_1}^1; B_{w_2}^1]) = \top$. Let also $\widetilde{tr} = tr'\cdot\mathsf{phase\ tall}\cdot bb(RB_{w_1})\cdot bb(RB_{w_2})$. By the semantics of our calculus, as well as Lemmas **??** and **??**, there must exist A'' such that $(\widetilde{tr}, A'') \in \mathsf{traces}(A_0^{n,2})$ and $\phi(A') = \phi(A'')$. Note that this implies, by compliance of Π_1 with $\{\mathsf{Test}^m\}_{m\in\mathbb{N}}$ and check, that $(tr'\cdot\mathsf{phase\ tall}\cdot bb(RB_{w_1})bb(RB_{w_2})\cdot\nu y.\overline{tal}\langle y\rangle, A''') \in \mathsf{traces}(A_0^{n_h,n_d,2})$ for some A'''.

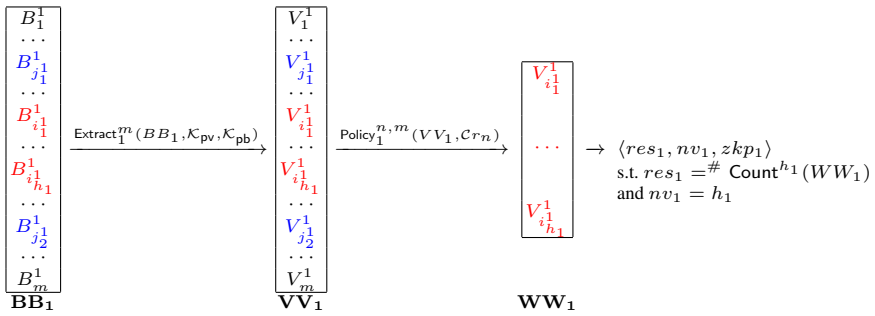
Let's now assume that there exists a process B''' such that $(\widetilde{tr}, B''') \in \mathsf{traces}(B_0^{n_h,n_d,2})$ with $\mathsf{check}([RB_{w_1}\phi(B'''); RB_{w_2}\phi(B''')]) = \top$. Then by the semantics of our calculus, as well as Lemmas **??** and **??**, there should exist B'' such that $(\widetilde{tr}, B'') \in \mathsf{traces}(B_0^{n_h,n_d,m})$ and $\phi(B'') = \phi(B''')$. In turn, this implies there exists B' such that $(\widehat{tr}, B') \in \mathsf{traces}(B_0^{n_h,n_d,m})$ and $\phi(B') = \phi(B'')$. Then this would mean that $\mathsf{check}([RB_{w_1}\phi(B'); RB_{w_2}\phi(B')])$, and by the second item of the definition of a property (Definition **??**) that $\mathsf{check}(BB_2) = \top$ for $BB_2 = [B_1^2; \ldots; B_m^2]$ and $B_i^2 =$

$RB_i\phi(B')$ for all $1 \leq i \leq m$. But we have already established that no such B' exists.

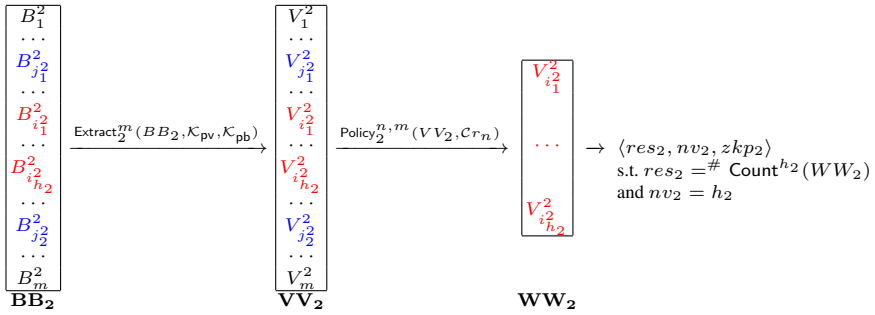
We have hence proved that $(tr' \cdot \mathsf{phase}\ \mathsf{tall} \cdot bb(RB_{w_1}) bb(RB_{w_2}) \cdot \nu y.\overline{tal}\langle y\rangle, A''') \in \mathsf{traces}(A_0^{n_h,n_d,2})$ is an attack that involves n voters but only two ballots tallied.

4.2. For some B, $(tr, B) \in \mathsf{traces}(B_0^{n_h,n_d,m})$. We know by Definition 6 that $y\phi(A) = \langle res_1, nvotes_1, zkp_1\rangle$, $y\phi(B) = \langle res_2, nvotes_2, zkp_2\rangle$, and thus by Lemma **??** that either $nvotes_1 \neq_{\mathsf{E}} nvotes_2$ or $res_1 \neq_{\mathsf{E}} res_2$.

Let $1 \leq i_1^1 < \cdots < i_{h_1}^1 \leq m$, $V_1^1, \ldots, V_m^1 \in \mathcal{C} \times Cr \cup \{\bot\}$, and $1 \leq j_1^1 < j_2^1 \leq m$ such that $\mathsf{check}([V_{j_1^1}^1; V_{j_2^1}^1]) = \top$ and



Let $1 \leq i_1^2 < \cdots < i_{h_2}^2 \leq m$, $V_1^2, \ldots, V_m^2 \in \mathcal{C} \times Cr \cup \{\bot\}$, and $1 \leq j_1^2 < j_2^2 \leq m$ such that $\mathsf{check}_2([V_{j_1^2}^2; V_{j_2^2}^2]) = \top$ and



Let $\{\ell_1, \ldots, \ell_h\} = \{i_1^1, \ldots, i_{h_1}^1\} \cup \{i_1^2, \ldots, i_{h_2}^2\} \cup \{j_1^1, j_2^1, j_1^2, j_2^2\}$. And, let $BB_1' = [B_{\ell_1}^1; \ldots; B_{\ell_h}^1]$, and $BB_2' = [B_{\ell_1}^2; \ldots; B_{\ell_h}^2]$.

- The first and last policies select at most n ballots, thus $\mathsf{Policy}_1^{n,m}$ and $\mathsf{Policy}_2^{n,m}$ select at most n ballots, so $1 \leq h_1, h_2 \leq n$. And thus $h \leq 4 + 2n$;
- Because Test^m is voting friendly (see Section 3.1), we know that $\mathsf{Test}^h(BB_1') = \mathsf{Test}^h(BB_2') = \top$;
- By definition of a property (Definition **??**), and because $\{j_1^1, j_2^1, j_1^2, j_2^2\} \subseteq \{\ell_1, \ldots, \ell_h\}$ with $\mathsf{check}([V_{j_1^1}^1; V_{j_2^1}^1]) = \mathsf{check}([V_{j_2^2}^2; V_{j_2^2}^2]) = \top$, we also have that $\mathsf{check}(BB_1') = \mathsf{check}(BB_2') = \top$;

– Because by assumption, $\mathsf{Extract}_1^m$ and $\mathsf{Extract}_2^m$ handle each ballot independently it is the case that $\mathsf{Extract}_1^h(BB'_1, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}) = [V^1_{\ell_1}; \ldots; V^1_{\ell_h}]$, and $\mathsf{Extract}_2^h(BB'_2, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}) = [V^2_{\ell_1}; \ldots; V^2_{\ell_h}]$.
– The first and last policies are stable by extract, and $\{i^1_1, \ldots, i^1_{h_1}\} \cup \{i^2_1, \ldots, i^2_{h_2}\} \subseteq \{\ell_1, \ldots, \ell_h\}$ with $\mathsf{Policy}_1^{n,m}(VV_1, \mathcal{C}r_n) = [V^1_{i^1_1}; \ldots; V^1_{i^1_{h_1}}]$ and $\mathsf{Policy}_2^{n,m}(VV_2, \mathcal{C}r_n) = [V^2_{i^2_1}; \ldots; V^2_{i^2_{h_2}}]$. Thus we know that $\mathsf{Policy}_1^{n,h}([V^1_{\ell_1}; \ldots; V^1_{\ell_h}], Cr_n) = [V^1_{i^1_1}; \ldots; V^1_{i^1_{h_1}}]$, and $\mathsf{Policy}_2^{n,h}([V^2_{\ell_1}; \ldots; V^2_{\ell_h}], Cr_n) = [V^2_{i^2_1}; \ldots; V^2_{i^2_{h_2}}]$.
– We can thus trivially conclude that $\langle res_1, nv_1, zkp'_1\rangle =^{\#} \mathsf{Tally}_1^{n,h}(BB'_1, Cr_n, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}})$, and $\langle res_2, nv_2, zkp'_2\rangle =^{\#} \mathsf{Tally}_2^{n,h}(BB'_2, Cr_n, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}})$ for some zkp'_1 and zkp'_2.

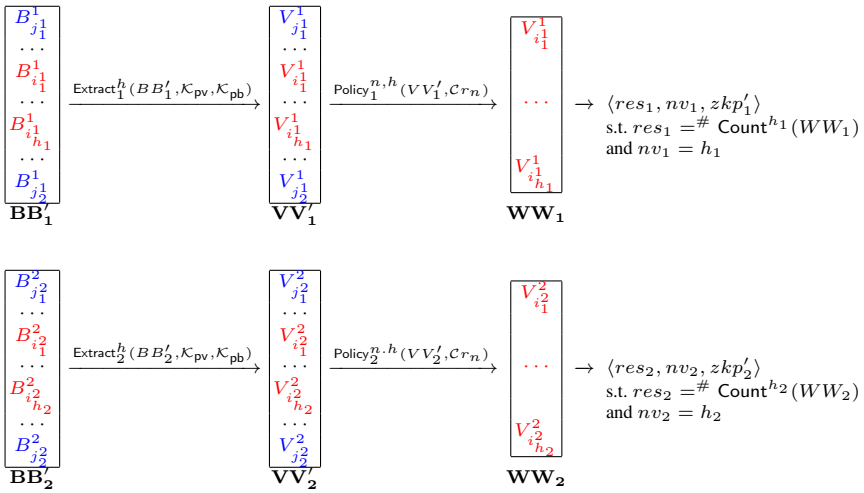
Now, Lemmas **??** and **??** imply that there exists C' and D' such that

$$(tr'\cdot\mathsf{phase}\ \mathsf{tall}\cdot bb(RB_{\ell_1})\ldots bb(RB_{\ell_h}), C') \in \mathsf{traces}(A_0^{n_h,n_d,h}) \text{ and } \phi(C') = \phi(A')$$
$$(tr'\cdot\mathsf{phase}\ \mathsf{tall}\cdot bb(RB_{\ell_1})\ldots bb(RB_{\ell_h}), D') \in \mathsf{traces}(B_0^{n_h,n_d,h}) \text{ and } \phi(D') = \phi(B')$$

And because our protocols comply with $\{\mathsf{Test}_m\}_{m\in\mathbb{N}}$ and check, that there exists C and D such that

$$(tr'\cdot\mathsf{phase}\ \mathsf{tall}\cdot bb(RB_{\ell_1})\ldots bb(RB_{\ell_h})\cdot \nu y.\overline{tal}\langle y\rangle, C) \in \mathsf{traces}(A_0^{n_h,n_d,h})$$
$$(tr'\cdot\mathsf{phase}\ \mathsf{tall}\cdot bb(RB_{\ell_1})\ldots bb(RB_{\ell_h})\cdot \nu y.\overline{tal}\langle y\rangle, D) \in \mathsf{traces}(B_0^{n_h,n_d,h})$$

with $y\phi(C) = \langle res_1, nv_1, zkp'_1\rangle$, and $y\phi(D) = \langle res_2, nv_2, zkp'_2\rangle$. In other words, we have established, that if we restrict the tallying to the ballots $\{\ell_1, \ldots, \ell_h\}$ the result will be the same as depicted by the following picture



Finally, because our protocols are almost determinate (Definition 5), we know that for all E such that $(tr'\cdot\mathsf{phase}\ \mathsf{tall}\cdot bb(RB_{\ell_1})\ldots bb(RB_{\ell_h})\cdot \nu y.\overline{tal}\langle y\rangle, E) \in \mathsf{traces}(B^{n_h,n_d,h})$, it must be that $y\phi(E) = \langle res''_2, nv_2, zkp''_2\rangle$ and that res''_2 is a permutation of res_2. But then we can conclude that for all processes E such that

$$(tr'\cdot\mathsf{phase}\ \mathsf{tall}\cdot bb(RB_{\ell_1})\ldots bb(RB_{\ell_h})\cdot \nu y.\overline{tal}\langle y\rangle, E) \in \mathsf{traces}(B_0^{n_h,n_d,h}),$$

then $C \not\sim E$.

Hence we have established that if there is an attack involving n voters (n_h honest and n_d dishonest ones), then there is one that involves at most $4 + 2n$ ballots tallied.

F Case study: the Helios protocol

Notation. To simplify the presentation of the case studies, we consider without loss of generality, elections with only two honest voters and any number of dishonest ones. From Lemma **??** we know that if there is an attack involving n_h honest voters and n_d dishonest ones, then there is an attack involving only 2 honest voters and the remaining $n_d + n_h - 2$ voters being dishonest. We will write $\Pi^{n,m}(\mathcal{C}r_n, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}})$ to denote $\Pi^{2,n-2,m}(\mathcal{C}r_2^h, \mathcal{C}r_{n-2}^d, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}})$ where $\mathcal{C}r_n = \mathcal{C}_2^h \cup \mathcal{C}_{n-2}^d$.

We consider several versions of the Helios protocol depending on the revoting policy, as well as on the mechanisms employed to enforce *ballot independence*. Ballot independence avoids in particular that a honest ballot can be copied by a dishonest voter which breaks privacy, see [13]. We also consider the two different tallying methods, namely mixnets-based and homomorphic tallying. At a high level, all these versions of Helios proceed as follows.

- The voter V computes her ballot by encrypting her vote with the public key $\mathsf{pk}(skE)$ of the election. The corresponding secret key is shared among several election authorities. Then, she casts her ballot and some auxiliary information on an authenticated channel. Depending on the version of Helios, this information may include her identity and a zero knowledge proof. Upon reception of the ballot, the administrator publishes the ballot and the auxiliary information on a public, append-only bulletin board.
- Once the voting phase is over, weeding may be applied to remove duplicate votes and a revote policy may remove ballots in case of multiple ballots correspond to same credentials.
- Next the ballots are tallied. In case of a mixnet based tally, the ballots are shuffled and their decryption is published together with a zero knowledge proof of correct decryption and mixing. In case of a homomorphic tally, the talliers homomorphically combine the encrypted votes and decrypt the result (with a zero knowledge proof of correct decryption).

A Helios election with n voters and m ballots to be tallied, can be modelled in our calculus by the process $H^{n,m}(\mathcal{C}r_n, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}, v_a, v_b)$ of the form:

$$
\begin{array}{l}
H^{n,m}(\mathcal{C}r_n, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}, v_a, v_b) \stackrel{\mathsf{def}}{=} \overline{bb}\langle \mathsf{pk}(skE)\rangle.\\
\qquad (1 : V(cred_a, id_a, v_a) \mid 1 : V(cred_b, id_b, v_b) \mid 2 : Tal^{n,m})\\
\mathcal{C}r_n \stackrel{\mathsf{def}}{=} \{(id_a, cred_a), (id_b, cred_b), \ldots, (id_n, cred_n)\}\\
\mathcal{K}_{\mathsf{pv}} \stackrel{\mathsf{def}}{=} \{skE\}\\
\mathcal{K}_{\mathsf{pb}} \stackrel{\mathsf{def}}{=} \{\mathsf{pk}(skE)\}
\end{array}
$$

This process represents two honests voters (id_a voting for candidate v_a and id_b voting for candidate v_b) and $n - 2$ dishonest voters under the intruder's control. We assume

identities to be publicly known. Hence we consider $\Phi_0 = \{xid_a \mapsto id_a, xid_b \mapsto id_b\} \cup \Phi$ for some Φ.

Several privacy properties are expressed as equivalences (like receipt-freeness or coercion-resistance). For simplicity, we focus here on vote privacy.

Although the homomorphic variants of the Helios protocol would also satisfy our hypotheses (and could thus benefit from our reduction results for their analysis), we did not prove that they satisfy ballot privacy, as ProVerif cannot handle the equational theory corresponding to a homomorphic encryption scheme. Therefore we focus on the mixnet variant of Helios in the remaining of this section.

F.1 With weeding and no revoting

One of the Helios variants ensures *ballot independance* simply by having the tallying authorities check that *(i)* encrypted ballot are pairwise distinct (this is also called weeding), and *(ii)* by checking the zero knowledge proof that ensures knowledge of the randomness used for encrypting the vote. This is captured by the family of public tests $\{\mathsf{Test}^m\}_{m\in\mathbb{N}}$ defined in Example 8. The *no revote policy* can be implemented by considering the first ballot casted by each voter as in the policy $\{\mathsf{Policy}^{n,m}_{\mathsf{first}}\}_{n,m\in\mathbb{N}}$ defined in Section 3.2. The extract function simply decrypts the encrypted vote and associates the voter's identity with it. This $\mathsf{Extract}$ function is formally defined in Example 9. Finally the count function depends on the tally. For a mixnet-based tally, this function is defined by the familly of functions $\{\mathsf{Count}^m_{DM}\}_{m\in\mathbb{N}}$ introduced in Example 10.

The last thing we need to define is the property check that captures that the honest voters have voted (their ballots are in the list of ballots to be tallied). For the versions of Helios where the identity is included in the ballot, we consider $\mathsf{check}([B_1,\ldots,B_m]) \stackrel{\text{def}}{=}$

$$\begin{cases} \top \text{ if } \exists i.\ \mathsf{Extract}(B_i) = (v_1, (id_a, cred_a)) \\ \quad \text{and } \exists j.\ \mathsf{Extract}(B_j) = (v_2, (id_b, cred_b)) \\ \quad \text{for some } v_1, v_2 \in \mathcal{V} \\ \bot \text{ otherwise} \end{cases}$$

Altogether we obtain a process modelling Helios with weeding, no revoting and mixnet-based tally, denoted by

$$\{H^{n,n}_{\mathsf{w\text{-}nr}}(\mathcal{C}r_n, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}, v_a, v_b)\}_{n\in\mathbb{N}}$$

The complete specification of this process may be found in our ProVerif file.

Proposition 3. *The mixnet-based Helios protocol with weeding and no revoting satisfies ballot privacy,* i.e. *for all $n \in \mathbb{N}$ such that $n \geq 2$*

$$H^{n,n}_{\mathsf{w\text{-}nr}}(\mathcal{C}r_n, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}, v_a, v_b) \approx H^{n,n}_{\mathsf{w\text{-}nr}}(\mathcal{C}r_n, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}, v_b, v_a)$$

Proof. We can easily check that $H^{n,n}_{\mathsf{w\text{-}nr}}$ is voting-friendly. Therefore. according to Theorem 1 it is sufficient to consider $n \in \{2, 3\}$. We used ProVerif to prove that in both cases the above equivalence is satisfied.

F.2 With weeding and revoting

When revoting is allowed, we consider the policy $\{\mathsf{Policy}^{n,m}_{\mathsf{last}}\}_{n,m\in\mathbb{N}}$ defined in Section 3.2. It is easy to encode this policy in the $Tal^{n,m}$ process. The resulting protocol modelling Helios with weeding and revoting is then denoted

$$\{H^{n,m}_{\mathsf{w\text{-}r}}(\mathcal{C}r_n, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}, v_a, v_b)\}_{m,n\in\mathbb{N}}$$

Proposition 4. *The mixnet-based Helios protocol with weeding and revoting satisfies ballot privacy,* i.e. *for all $n, m \in \mathbb{N}$ such that $n \geq 2$*

$$H^{n,m}_{\mathsf{w\text{-}r}}(\mathcal{C}r_n, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}, v_a, v_b) \approx H^{n,m}_{\mathsf{w\text{-}r}}(\mathcal{C}r_n, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}, v_b, v_a)$$

Proof. Again, we can easily check that $H^{n,n}_{\mathsf{w\text{-}r}}$ is voting-friendly. Since each voter sends her credentials (her identity) in plaintext, and since the $\mathsf{Extract}$ function associates this identity with the submitted vote without further verification, we can conclude that the Helios protocol has identifiable ballots. Now according to Theorem 3 it is sufficient to consider $n \in \{2, 3\}$ and $m \in \{1, \ldots, 7\}$. We used ProVerif to prove that in any case the above equivalence is satisfied.

F.3 With identities in zero knowledge proofs and no re-voting

A more efficient way of enforcing *ballot independence* is by including the identity of the voter in the zero knowledge proof demonstrating the knowledge of the encryption randomness. This assumes a slightly different equational theory for the zero knowledge proofs sent by the voters

$$\mathsf{checkzkp}_{\mathsf{E}}(\mathsf{zkp}_{\mathsf{E}}(xr, xv, xid, \mathsf{aenc}(xpk, xr, xv)), xid, \\ \mathsf{aenc}(xpk, xr, xv)) = \mathsf{okzkp}_{\mathsf{E}}$$

and adapting the zero knowledge proof in the ballot to include the identity. The family of test functions does now not require the global weeding test anymore

$$Test^m([B_1, \ldots, B_m]) \stackrel{\text{def}}{=} \bigwedge_{i=1}^{i=m} \mathsf{lTest}(B_i)$$

$$\mathsf{lTest}(B) \stackrel{\text{def}}{=} \begin{cases} \top \text{ if } B = \langle id, bal, prf\rangle \\ \quad \text{and } \mathsf{checkzkp}_{\mathsf{E}}(prf, id, \mathsf{getmsg}(bal)) = \mathsf{okzkp}_{\mathsf{E}} \\ \bot \text{ otherwise} \end{cases}$$

The resulting protocol modelling Helios with identities in zero knowledge proofs and no revoting is denoted

$$\{H^{n,n}_{\mathsf{s\text{-}nr}}(\mathcal{C}r_n, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}, v_a, v_b)\}_{n\in\mathbb{N}}$$

Proposition 5. *The mixnet-based Helios protocol with identities in zero knowledge proofs and no revoting satisfies ballot privacy,* i.e. *for all $n \in \mathbb{N}$ such that $n \geq 2$*

$$H^{n,n}_{\mathsf{id\text{-}nr}}(\mathcal{C}r_n, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}, v_a, v_b) \approx H^{n,n}_{\mathsf{id\text{-}nr}}(\mathcal{C}r_n, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}, v_b, v_a)$$

Proof. Since $H^{n,n}_{\mathsf{id\text{-}nr}}$ is voting-friendly and according to Theorem 1 it is sufficient to consider $n \in \{2, 3\}$. We used ProVerif to prove that in both cases the above equivalence is satisfied.

F.4 With identities in zero knowledge proofs and re-voting

The variant of Helios that relies on identities in zero knowledge proofs to enforce *ballot independence* and allows revoting is also in the scope of our reduction results. We denote this version of the protocol by

$$\{H^{n,m}_{\mathsf{id\text{-}r}}(\mathcal{C}r_n, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}, v_a, v_b)\}_{m,n\in\mathbb{N}}$$

Proposition 6. *The decryption mixnets-based Helios protocol with signatures and revoting satisfies ballot privacy,* i.e. *for all* $n, m \in \mathbb{N}$ *such that* $n \geq 2$

$$H^{n,m}_{\mathsf{id\text{-}r}}(\mathcal{C}r_n, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}, v_a, v_b) \approx H^{n,m}_{\mathsf{id\text{-}r}}(\mathcal{C}r_n, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}, v_b, v_a)$$

Proof. Because $H^{n,m}_{\mathsf{id\text{-}r}}$ has identifiable ballots and is voting-friendly, it is sufficient according to to Theorem 3 to consider $n \in \{2, 3\}$ and $m \in \{1, \ldots, 7\}$. We used ProVerif to prove that in any case the above equivalence is satisfied.

F.5 Belenios

One limitation of Helios is that a dishonest bulletin board could perform ballot stuffing for eligible voters that did not cast any vote. To overcome this problem, Cortier et al. [19] proposed the Belenios system which builds on Helios. The main difference is that the system relies on an additional authority, the credential issuer, which is assumed not to collaborate with the bulletin board. The credential issuer distributes to each eligible voter a signature key pair and publishes the list of public keys, which constitues the list of eligible voters.

We can easily adapt our Helios model to Belenios and apply our result, yielding a bound of 3 voters and a total of 3 ballots when no revoting is allowed and a total of 7 ballots otherwise.

Ballots are now signed with the private credential and the zero knowledge proof also depends on the credential, instead of the identity. Signatures are straightforwardly modeled by the equation

$$\mathsf{checksig}(\mathsf{vk}(xk), \mathsf{sig}(xk, xm), xm) = \mathsf{oksig}$$

and the ballot is of the form $B = \langle \mathsf{vk}(sk), bal, prf, sig\rangle$ where

$$\begin{aligned} bal &= \mathsf{aenc}(\mathsf{pk}(skE), r, v)\\ prf &= \mathsf{zkp}(r, v, \mathsf{vk}(sk), \mathsf{aenc}(\mathsf{pk}(skE), r, v))\\ sig &= \mathsf{sig}(sk, \langle \mathsf{vk}(sk), bal, prf\rangle) \end{aligned}$$

for some randomness r. The tests performed by the tallying authority and the extract functions are also updated accordingly. The resulting protocol modelling Belenios with either no revoting or revoting are denoted by

$$\{B^{n,n}_{\mathsf{nr}}(\mathcal{C}r_n, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}, v_a, v_b)\}_{n\in\mathbb{N}}$$

respectively,

$$\{B^{n,m}_{\mathsf{r}}(\mathcal{C}r_n, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}, v_a, v_b)\}_{m,n\in\mathbb{N}}$$

Proposition 7. *The Belenios protocol with no revoting, respectively revoting, satisfies ballot privacy,* i.e. *for all $n, m \in \mathbb{N}$ such that $n \geq 2$*

$$B^{n,n}_{\mathsf{nr}}(\mathcal{C}r_n, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}, v_a, v_b) \approx B^{n,n}_{\mathsf{nr}}(\mathcal{C}r_n, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}, v_b, v_a)$$

and

$$B^{n,m}_{\mathsf{r}}(\mathcal{C}r_n, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}, v_a, v_b) \approx B^{n,m}_{\mathsf{r}}(\mathcal{C}r_n, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}, v_b, v_a)$$

Proof. The proof is done using ProVerif relying on Theorems 1 and 3 as for other versions of Helios.

G Case study: the Juels, Catalano, and Jakobsson (JCJ) protocol

The JCJ protocol relies on *anonymous credentials* to enforce privacy. They simply consist of random values encrypted under the election authorities public key. Legitimacy of ballots is ensured by mixing and blind comparisons (*plaintext equivalence tests*) between credentials used for casting ballots and the list of legitimate ones. This allows authorities to check the presence of a *concealed credential* in a list of valid ones. Such anonymous credentials, further allow for a voter under coercion to transmit a *fake credential* to her coercer, without the coercer being able to tell whether the credential is valid or not. The following equations capture the properties of the encryption scheme and zero knowledge proofs used in JCJ:

$$\begin{array}{l}
\mathsf{adec}(xsk, \mathsf{aenc}(\mathsf{pub}(xsk), xr, xm)) = xm \\
\mathsf{pet}(xsk, \mathsf{aenc}(\mathsf{pub}(xsk), xr_1, xm), \mathsf{aenc}(\mathsf{pub}(xsk), xr_2, xm)) = \mathsf{okpet} \\
\mathsf{checkzkp}(\\
\quad \mathsf{zkp}(xrv, xrc, xv, xc, \mathsf{aenc}(xpkT, xrv, xv), \mathsf{aenc}(xpkR, xrc, xc)), \\
\quad xpkT, \mathsf{aenc}(xpkT, xrv, xv), \\
\quad xpkR, \mathsf{aenc}(xpkR, xrc, xc)) = \mathsf{okzkp}
\end{array}$$

For an election with n voters, the set of credentials is

$$\begin{array}{rl}
\mathcal{C}r_n \stackrel{\mathsf{def}}{=} & \{\mathsf{aenc}(\mathsf{pub}(skR), r_a, c_a), \mathsf{aenc}(\mathsf{pub}(skR), r_b, c_b), \\
& \mathsf{aenc}(\mathsf{pub}(skR), r_3, c_3), \ldots, \mathsf{aenc}(\mathsf{pub}(skR), r_n, c_n)\}
\end{array}$$

where skR denotes the secret key of the registrars. The encrypted credentials of the two honest voters is made public, as well as the public keys of the registrars and talliers. So we consider

$$\begin{array}{rl}
\mathcal{E}_0 \stackrel{\mathsf{def}}{=} & \{skT, skR, c_a, r_a, c_b, r_b\} \\
\Phi_0 \stackrel{\mathsf{def}}{=} & \{xpkT \mapsto \mathsf{pub}(skT), xpkR \mapsto \mathsf{pub}(skR), \\
& \quad xc_a \mapsto \mathsf{aenc}(\mathsf{pub}(skR), r_a, c_a), \\
& \quad xc_b \mapsto \mathsf{aenc}(\mathsf{pub}(skR), r_b, c_b)\}
\end{array}$$

To cast a ballot, a voter encrypts her choice of candidate with the talliers' public key. She also encrypts her credential with the registrars' public key, and builds a zero knowledge proof to demonstrate knowledge of the randomness used in these two encryptions. She then sends to the talliers, her encrypted vote, together with her encrypted credential and the corresponding proof.

Once the casting phase is over, the talliers proceed as follow. For each ballot they check the accompanying zero knowledge proof. This is captured by the following family of public tests

$$Test^m([B_1,\ldots,B_m]) \stackrel{\text{def}}{=} \bigwedge_{i=1}^{i=m} \mathsf{ITest}(B_i)$$

$$\mathsf{ITest}(B) \stackrel{\text{def}}{=} \begin{cases} \top \text{ if } B = \langle ev, ec, prf\rangle \\ \quad \text{and } \mathsf{checkzkp}(prf, \mathsf{pub}(skT), ev, \\ \qquad\qquad \mathsf{pub}(skR), ec) =_{\mathsf{E}} \mathsf{okzkp} \\ \bot \text{ otherwise} \end{cases}$$

They then determine the set of valid votes. That is the votes submitted with a valid credential. This is captured by the following Extract function

$$\mathsf{Extract}(B, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}) \stackrel{\text{def}}{=} \begin{cases} \top \text{ if } B = \langle ev, ec, prf\rangle \\ \quad \text{and } \mathsf{pet}(skR, ev, ev' =_{\mathsf{E}} \mathsf{okpet} \\ \quad \text{for some } ev' \in \mathcal{C}r_n \\ \bot \text{ otherwise} \end{cases}$$

where $\mathcal{K}_{\mathsf{pv}} = \{skT, skR\}$ and $\mathcal{K}_{\mathsf{pb}} = \{\mathsf{pub}(skT), \mathsf{pub}(skR)\}$. The Extract function performs a plaintext equivalence test between the submitted encrypted credential and the list of legitimate encrypted credentials. If this test succeeds for one of the valid credentials then the ballot is considered to be valid.

The revoting policy of JCJ retains only the last vote, which is formalised by the family of functions $\{\mathsf{Policy}^{n,m}_{\mathsf{last}}\}_{n,m\in\mathbb{N}}$, introduced in Section 3.2. Finally, the tallying authorities publish the set of valid votes in a random order. This is captured by the family of count functions $\{\mathsf{Count}^m_{DM}\}_{m\in\mathbb{N}}$ introduced in Example 10.

The last thing we need to define is the check check that captures that the honest voters have voted (their ballots are in the list of ballots to be tallied)

$$\mathsf{check}([B_1,\ldots,B_m]) \stackrel{\text{def}}{=} \begin{cases} \top \text{ if } \exists i.\ \mathsf{Extract}(B_i) = (cr_a, v_1) \\ \quad \text{and } \exists j.\ \mathsf{Extract}(B_j) = (cr_b, v_2) \\ \quad \text{where } cr_a = \mathsf{aenc}(\mathsf{pub}(skR), r_a, c_a) \\ \quad \text{and } cr_b = \mathsf{aenc}(\mathsf{pub}(skR), r_a, c_a) \\ \quad \text{for some } v_1, v_2 \in \mathcal{V} \\ \bot \text{ otherwise} \end{cases}$$

The resulting family of processes modelling the JCJ protocol is denoted

$$\{JCJ^{n,m}(\mathcal{C}r_n, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}, v_a, v_b)\}_{n\in\mathbb{N}}.$$

To prove that the JCJ protocol satisfies ballot privacy, the following equivalence would need to be established. For all $n, m \in \mathbb{N}$ such that $n \geq 2$

$$JCJ^{n,m}(\mathcal{C}r_n, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}, v_a, v_b) \approx JCJ^{n,m}(\mathcal{C}r_n, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}, v_b, v_a)$$

$JCJ^{n,m}$ is voting friendly thus according to Theorem 2 it would be sufficient to consider $n \in \{2,3\}$ and $m \in \{1,\ldots,10\}$. Unfortunately, due to state explosion,

ProVerif is not able to prove this equivalence for $m \geq 3$. This may sound surprising since Backes et al. [11] conducted an analysis of JCJ for an arbitrary numer of voters. The major difference with our approach is that they consider anonymous channels, that is, they assume that voters communicate their ballots anonymously to the ballot box, which considerably simplifies the model. While anonymous channels are indeed assumed in the original presentation of the JCJ protocol to avoid forced abstention, it is not needed for vote secrecy and anonymous channels are not part of the the Civitas system [29], implementing JCJ. As it seems unreasonable to assume that most voters' connections are anonymous, we analyse here a more practical variant of JCJ, without anonymous channels. Unsurprisingly, it is harder to prove vote secrecy without this assumption.

H Case study: the Prêt-à-Voter (PaV) protocol

The main idea of PaV is that an election authority creates ballots that contain the names of the candidates in a random order on the left-hand side and their corresponding encryption in the same order on the right-hand side. This allows the voter to mark a vote for the desired candidate and scan only the encrypted part of the ballot, the right-hand side, to be posted on the bulletin board. The random order of candidates in the ballot and the decryption key are assumed to be secret. Moreover, ballot forms are assumed to be unforgeable, ensuring vote privacy, and even coercion-resistance, if care is taken to destroy the left-hand side.

After the encrypted votes get to the bulletin board, the design of PaV is similar to other voting systems like JCJ/Civitas or Helios: ballots are either decrypted by a decryption mixnet ensuring privacy; or are first anonymized by a re-encryption mixnet and then decrypted by the owners of the secret key.

Unforgeability of paper ballots can be captured in a formal model by *(i)* having the talliers check that all submitted ballots are different, but also *(ii)* relying on a zero knowledge proof demonstrating knowledge of the randomness used to create the ballot. We can in that case consider the family of public tests $\{\mathsf{Test}^m\}_{m\in\mathbb{N}}$ defined in Example 8. As PaV does not allow revoting, the policy can just select the first submitted vote by each voter. This is captured by the family of policy functions $\{\mathsf{Policy}^{n,m}_{\mathsf{first}}\}_{n,m\in\mathbb{N}}$ described in Section 3.2. The Extract function simply decrypts the encrypted vote, as specified at Example 9; and the mixnets-based tallying of votes is captured by the family of functions $\{\mathsf{Count}^m_{DM}\}_{m\in\mathbb{N}}$ introduced in Example 10. Ballot privacy is captured by the same check as for the Helios protocol (see Section **??**).

The resulting protocol modelling decryption mixnets-based tallying is denoted

$$\{PaV^{n,n}_{\mathsf{DM}}(\mathcal{C}r_n, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}, v_a, v_b)\}_{n\in\mathbb{N}}$$

It satisfies our assumption, so 3 voters and 3 ballots are sufficient.

Proposition 8. *The decryption mixnets-based PaV protocol satisfies ballot privacy,* i.e. *for all $n \in \mathbb{N}$ such that $n \geq 2$*

$$PaV^{n,n}_{\mathsf{DM}}(\mathcal{C}r_n, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}, v_a, v_b) \approx PaV^{n,n}_{\mathsf{DM}}(\mathcal{C}r_n, \mathcal{K}_{\mathsf{pv}}, \mathcal{K}_{\mathsf{pb}}, v_b, v_a)$$

Proof. According to Theorem 1 it is sufficient to consider $n \in \{2, 3\}$. We used ProVerif to prove that in both cases the above equivalence is satisfied.

Note that this yields the first proof of vote privacy for the decryption mixnets-based PaV protocol.

The reencryption mixnets-based variant (RM) of the PaV protocol also satisfy our hypotheses (and could thus benefit from our reduction results for their analysis). But, we did not prove that it satisfies ballot privacy, as ProVerif cannot handle the equational theory corresponding to reencryption.