D4-2: Results on case studies from literature

Steve Kremer

LSV, CNRS & ENS Cachan & INRIA

The results presented in this report are based on joint work with S. Delaune, M. Ryan and B. Smyth

In this report we give an overview of our work on analyzing e-voting protocols from the literature. In [DKR09], we have analysed the privacy-type properties for three protocols: Fujioka et al. [FOO92], Okamaoto [Oka96] and Lee et al. [LBD+04]. In [KRS10], we have analysed verifiability properties of the protocols by Fujioka et al. [FOO92] and Juels et al. [JCJ05] recently implemented as Civitas [CCM08]. For each of these two families of properties we summarize our formal model of the protocols and the properties as well as the results on the case studies. More detailed information can be found in the full papers which are appended to this report.

1 Privacy-type properties

In this section we describe our work on privacy type properties. We distinguish three types of privacy properties:

- Vote-privacy: the fact that a particular voter voted in a particular way is not revealed to anyone.
- Receipt-freeness: a voter does not gain any information (a *receipt*) which can be used to prove to a coercer that she voted in a certain way.
- Coercion-resistance: a voter cannot cooperate with a coercer to prove to him that she voted in a certain way.

The weakest of the three, called *vote-privacy*, roughly states that the fact that a voter voted in a particular way is not revealed to anyone. When stated in this simple way, however, the property is in general false, because if all the voters vote unanimously then everyone will get to know how everyone else voted. The formalisation we give in fact says that no party receives information which would allow them to distinguish one situation from another one in which two voters swap their votes.

Receipt-freeness says that the voter does not obtain any artefact (a "receipt") which can be used later to prove to another party how she voted. Such a receipt may be intentional or unintentional on the part of the designer of the system. Unintentional receipts might include nonces or keys which the voter is given during the protocol. Receipt-freeness is a stronger property than privacy. Intuitively, privacy says that an attacker cannot discern how a voter votes from any information that the voter necessarily reveals during the course of the election. Receipt-freeness says the same thing even if the voter voluntarily reveals additional information.

Coercion-resistance is the third and strongest of the three privacy properties. Again, it says that the link between a voter and her vote cannot be established by an attacker, this time even if the voter cooperates with the attacker during the election process. Such cooperation can include giving to the attacker any data which she gets during the voting process, and using data which the attacker provides in return. When analysing coercion-resistance, we assume that the voter and the attacker can communicate and exchange data at any time during the election process. Coercion-resistance is intuitively stronger than receipt-freeness, since the attacker has more capabilities.

Of course, the voter can simply tell an attacker how she voted, but unless she provides convincing evidence the attacker has no reason to believe her. Receipt-freeness and coercion-resistance assert that she cannot provide convincing evidence. Coercion-resistance cannot possibly hold if the coercer can physically vote on behalf of the voter. Some mechanism is necessary for isolating the voter from the coercer at the moment she casts her vote. This can be realised by a voting booth,

which we model here as a private and anonymous channel between the voter and the election administrators.

Note that in literature the distinction between receipt-freeness and coercion-resistance is not very clear. The definitions are usually given in natural language and are insufficiently precise to allow comparison. The notion of receipt-freeness first appeared in the work of Benaloh and Tuinstra [BT94]. Since then, several schemes [BT94,Oka96] were proposed in order to meet the condition of receipt-freeness, but later shown not to satisfy it. One of the reasons for such flaws is that no formal definition of receipt-freeness has been given. The situation for coercion-resistance is similar. Systems have been proposed aiming to satisfy it; for example, Okamoto [Oka97] presents a system resistant to interactive coercers, thus aiming to satisfy what we call coercion-resistance, but this property is stated only in natural language. A rigorous definition in a computational model has been proposed by Juels *et al.* for coercion-resistance [JCJ05] and in the UC framework by Moran and Naor [MN06] and Unruh and Müller-Quade [UM10]. To the best of our knowledge our definition is the first "formal methods" definition of receipt-freeness and coercion-resistance. It is difficult to compare our definition and the ones proposed in [JCJ05,MN06,UM10] due to the inherently different models. Our work has later been extended by Backes *et al.* [BHM08] who aim automation of coercion-resistance using ProVerif.

This section is based on the results published in [DKR09].

1.1 Formalising voting protocols

Before formalising security properties, we need to define what is an electronic voting protocol in the applied pi calculus. Different voting protocols often have substantial differences. However, we believe that a large class of voting protocols can be represented by processes corresponding to the following structure.

Definition 1 (Voting process). *A voting process is a closed plain process*

$$VP \equiv \nu\tilde{n}.(V\sigma_1 \mid \cdots \mid V\sigma_n \mid A_1 \mid \cdots \mid A_m).$$

The $V\sigma_i$ are the voter processes, the A_js the election authorities which are required to be honest and the $\tilde{n}$ are channel names. We also suppose that $v \in \mathsf{dom}(\sigma_i)$ is a variable which refers to the value of the vote. We define an evaluation context S which is as VP, but has a hole instead of two of the $V\sigma_i$.

In order to prove a given property, we may require some of the authorities to be honest, while other authorities may be assumed to be corrupted by the attacker. The processes $A_1, \ldots, A_m$ represent the authorities which are required to be honest. The authorities under control of the attacker need not be modelled, since we consider any possible behaviour for the attacker (and therefore any possible behaviour for corrupt authorities). In this case the communication channels are available to the environment.

1.2 Vote-privacy

The privacy property aims to guarantee that the link between a given voter V and his vote v remains hidden. Anonymity and privacy properties have been successfully studied using equivalences, *e.g.* [SS96] . However, the definition of privacy in the context of voting protocols is rather subtle. While generally most security properties should hold against an arbitrary number of dishonest participants, arbitrary coalitions do not make sense here. Consider for instance the case where all but one voter are dishonest: as the results of the vote are published at the end, the dishonest voter can collude and determine the vote of the honest voter. A classical trick for modelling anonymity is to ask whether two processes, one in which V_A votes and one in which V_B votes, are equivalent. However, such an equivalence does not hold here as the voters' identities are revealed (and they need to be revealed at least to the administrator to verify eligibility). In a similar way, an equivalence of two processes where only the vote is changed does not hold, because the votes

are published at the end of the protocol. To ensure privacy we need to hide the *link* between the voter and the vote and not the voter or the vote itself.

In order to give a reasonable definition of privacy, we need to suppose that at least two voters are honest. We denote the voters V_A and V_B and their votes a, respectively b. We say that a voting protocol respects privacy whenever a process where V_A votes a and V_B votes b is observationally equivalent to a process where V_A votes b and V_B votes a. Formally, privacy is defined as follows.

Definition 2 (Vote-privacy). *A voting protocol respects* vote-privacy *(or just* privacy*) if*

$$S[V_A\{^a/_v\} \mid V_B\{^b/_v\}] \approx_\ell S[V_A\{^b/_v\} \mid V_B\{^a/_v\}]$$

for all possible votes a and b.

The intuition is that if an intruder cannot detect if arbitrary honest voters V_A and V_B swap their votes, then in general he cannot know anything about how V_A (or V_B) voted. Note that this definition is robust even in situations where the result of the election is such that the votes of V_A and V_B are necessarily revealed. For example, if the vote is unanimous, or if all other voters reveal how they voted and thus allow the votes of V_A and V_B to be deduced.

As already noted, in some protocols the vote-privacy property may hold even if authorities are corrupt, while other protocols may require the authorities to be honest. When proving privacy, we choose which authorities we want to model as honest, by including them in Definition 1 of *VP* (and hence S).

1.3 Receipt-Freeness

Similarly to privacy, receipt-freeness may be formalised as an observational equivalence. We also formalise receipt-freeness using observational equivalence. However, we need to model the fact that V_A is willing to provide secret information, i.e., the receipt, to the coercer. We assume that the coercer is in fact the attacker who, as usual in the Dolev-Yao model, controls the public channels. To model V_A's communication with the coercer, we consider that V_A executes a voting process which has been modified. We denote by P^{ch} the plain process P that is modified as follows: any input of base type and any freshly generated names of base type are output on channel ch. We do not forward restricted channel names, as these are used for modelling purposes, such as physically secure channels, *e.g.* the voting booth, or the existence of a PKI which securely distributes keys (the keys themselves are forwarded but not the secret channel name on which the keys are received). In the remainder, we assume that $ch \notin fn(P) \cup bn(P)$ before applying the transformation. Given an extended process A and a channel name ch, we to define the extended process $A^{\setminus out(ch,\cdot)}$ as $\nu ch.(A \mid !\mathsf{in}(ch, x))$. Intuitively, such a process is as the process A, but hiding the outputs on the channel ch.

We are now ready to define receipt-freeness. Intuitively, a protocol is receipt-free if, for all voters V_A, the process in which V_A votes according to the intruder's wishes is indistinguishable from the one in which she votes something else. As in the case of privacy, we express this as an observational equivalence to a process in which V_A swaps her vote with V_B, in order to avoid the case in which the intruder can distinguish the situations merely by counting the votes at the end. Suppose the coercer's desired vote is c. Then we define receipt-freeness as follows.

Definition 3 (Receipt-freeness). *A voting protocol is* receipt-free *if there exists a closed plain process V' such that*

- $V'^{\setminus out(chc,\cdot)} \approx_\ell V_A\{^a/_v\}$,
- $S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}] \approx_\ell S[V' \mid V_B\{^c/_v\}]$,

for all possible votes a and c.

As before, the context S in the second equivalence includes those authorities that are assumed to be honest. V' is a process in which voter V_A votes a but communicates with the coercer C in order to feign cooperation with him. Thus, the second equivalence says that the coercer cannot tell the difference between a situation in which V_A genuinely cooperates with him in order to cast the vote c and one in which she pretends to cooperate but actually casts the vote a, provided there is some counter-balancing voter that votes the other way around. The first equivalence of the definition says that if one ignores the outputs V' makes on the coercer channel chc, then V' looks like a voter process V_A voting a.

The first equivalence of the definition may be considered too strong; informally, one might consider that the equivalence should be required only in a particular S context rather than requiring it in any context (with access to all the private channels of the protocol). This would result in a weaker definition, although one which is more difficult to work with. In fact, the variant definition would be only slightly weaker; it is hard to construct a natural example which distinguishes the two possibilities, and in particular it makes no difference to the case studies of later sections. Therefore, we prefer to stick to Definition 3.

According to intuition, if a protocol is receipt-free (for a given set of honest authorities), then it also respects privacy (for the same set):

Proposition 1. *If a voting protocol is receipt-free then it also respects privacy.*

1.4 Coercion-Resistance

Coercion-resistance is a stronger property as we give the coercer the ability to communicate *interactively* with the voter and not only receive information. In this model, the coercer can prepare the messages he wants the voter to send. As for receipt-freeness, we modify the voter process. In the case of coercion-resistance, we give the coercer the possibility to provide the messages the voter should send. The coercer can also decide how the voter branches on *if*-statements.

We denote by P^{c_1,c_2} the plain process P that is modified as follows: any input of base type and any freshly generated names of base type are output on channel c_1. Moreover, when M is a term of base type, any output $\mathsf{out}(u, M)$ is replaced by $\mathsf{in}(c_2, x).\mathsf{out}(u, x)$ where x is a fresh variable and any occurence of if $M = N$ is replaced by $\mathsf{in}(c_2, x)$. if $x = \text{true}$.

As a first approximation, we could try to define coercion-resistance in the following way: a protocol is coercion-resistant if there is a V' such that

$$S[V_A\{^{?}/_{v}\}^{c_1,c_2} \mid V_B\{^{a}/_{v}\}] \approx_\ell S[V' \mid V_B\{^{c}/_{v}\}]. \tag{1}$$

On the left, we have the coerced voter $V_A\{^{?}/_{v}\}^{c_1,c_2}$; no matter what she intends to vote (the "?"), the idea is that the coercer will force her to vote c. On the right, the process V' resists coercion, and manages to vote a. Unfortunately, this characterisation has the problem that the coercer could oblige $V_A\{^{?}/_{v}\}^{c_1,c_2}$ to vote $c' \neq c$. In that case, the process $V_B\{^{c}/_{v}\}$ would not counter-balance the outcome to avoid a trivial way of distinguishing the two sides.

To enable us to reason about the coercer's choice of vote, we model the coercer's behaviour as a context C that defines the interface c_1, c_2 for the voting process. The context C coerces a voter to vote c. Thus, we can characterise coercion-resistance as follows: a protocol is coercion-resistant if there is a V' such that

$$S[C[V_A\{^{?}/_{v}\}^{c_1,c_2}] \mid V_B\{^{a}/_{v}\}] \approx_\ell S[C[V'] \mid V_B\{^{c}/_{v}\}], \tag{2}$$

where C is a context ensuring that the coerced voter $V_A\{^{?}/_{v}\}^{c_1,c_2}$ votes c. The context C models the coercer's behaviour, while the environment models the coercer's powers to observe whether the coerced voter behaves as instructed. We additionally require that the context C does not directly use the channel names $\tilde{n}$ restricted by S. Formally one can ensure that $V_A\{^{?}/_{v}\}^{c_1,c_2}$ votes c by requiring that $C[V_A\{^{?}/_{v}\}^{c_1,c_2}] \approx_\ell V_A\{^{c}/_{v}\}^{chc}$. We actually require a slightly weaker condition, $S[C[V_A\{^{?}/_{v}\}^{c1,c2}] \mid V_B\{^{a}/_{v}\}] \approx_\ell S[V_A\{^{c}/_{v}\}^{chc} \mid V_B\{^{a}/_{v}\}]$, which results in a stronger property. Backes *et al.* [BHM08] propose a variant of our definitions: instead of forcing the coercer's vote

to c, they require the existence of an *extractor* process which extracts the vote of the coercer to enable counter-balancing.

Putting the above ideas together, we get to the following definition:

Definition 4 (Coercion-resistance). *A voting protocol is* coercion-resistant *if there exists a closed plain process V' such that for any $C = \nu c_1.\nu c_2.(\_ \mid P)$ satisfying $\tilde{n} \cap fn(C) = \emptyset$ and $S[C[V_A\{^?/_v\}^{c1,c2}] \mid V_B\{^a/_v\}] \approx_\ell S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}]$, we have*

- $C[V']^{\setminus out(chc,\cdot)} \approx_\ell V_A\{^a/_v\}$,
- $S[C[V_A\{^?/_v\}^{c_1,c_2}] \mid V_B\{^a/_v\}] \approx_\ell S[C[V'] \mid V_B\{^c/_v\}]$.

Note that $V_A\{^?/_v\}^{c_1,c_2}$ does not depend on what we put for "?".

The condition that $S[C[V_A\{^?/_v\}^{c1,c2}] \mid V_B\{^a/_v\}] \approx_\ell S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}]$ means that the context C outputs the secrets generated during its computation; this is required so that the environment can make distinctions on the basis of those secrets, as in receipt-freeness. The first bullet point expresses that V' is a voting process for A which fakes the inputs/outputs with C and succeeds in voting a in spite of the coercer. The second bullet point says that the coercer cannot distinguish between V' and the really coerced voter, provided another voter V_B counter-balances.

As in the case of receipt-freeness, the first equivalence of the definition could be made weaker by requiring it only in a particular S context. But we chose not to adopt this extra complication, for the same reasons as given in the case of receipt-freeness.

Remark 1. The context C models the coercer's behaviour; we can see its role in equivalence (2) as imposing a restriction on the distinguishing power of the environment in equivalence (1). Since the coercer's behaviour is modelled by C while its distinguishing powers are modelled by the environment, it would be useful to write (2) as

$$C[S[V_A\{^?/_v\}^{c_1,c_2}] \mid V_B\{^a/_v\}]] \approx_\ell C[S[V' \mid V_B\{^c/_v\}]]. \tag{3}$$

We have shown that equivalences (2) and (3) are the same.

Remark 2. Note that our definition of coercion-resistance cannot cover attacks such as the *ballot-as-signature attack* (also known as the *Italian attack*) [DC] where the number of possible votes is extremely high and therefore a particular vote is unlikely to appear twice and can therefore be identified by a coercer.

According to intuition, if a protocol is coercion-resistant then it respects receipt-freeness too (as before, we keep constant the set of honest authorities):

Proposition 2. *If a voting protocol is coercion-resistant then it also respects receipt-freeness.*

1.5 Case studies

We have analysed the above discussed privacy-type properties for three protocols: Fujioka et al. [FOO92], Okamaoto [Oka96] and Lee et al. [LBD+04]. As we only model authorities that are required to be honest for these protocols to hold we are able to identify which authorities need to be trusted for these particular properties. When analysing these three properties the existence of the process V' for receipt-freeness and coercion-resistance turned out to be easy. In the protocol specification these processes are generally described as the way of achieving the properties. The equivalence properties could however not be proved automatically and required hand proofs. We were however able to rely on ProVerif for some Lemmas on static equivalence. We summarise the results of these three case studies in Figure 1.

Property	*Fujioka et al.*	*Okamoto*	*Lee et al.*
Vote-privacy	✓	✓	✓
trusted authorities	none	timeliness mbr.	administrator
Receipt-freeness	×	✓	✓
trusted authorities	n/a	timeliness mbr.	admin. & collector
Coercion-resistance	×	×	✓
trusted authorities	n/a	n/a	admin. & collector

Fig. 1. Summary of protocols and properties

2 Election verifiability

We present a definition of election verifiability which captures three desirable aspects: individual, universal and eligibility verifiability. We formalise verifiability as a triple of Boolean tests $\Phi^{\mathsf{IV}}, \Phi^{\mathsf{UV}}, \Phi^{\mathsf{EV}}$ which are required to satisfy several conditions on all possible executions of the protocol. Φ^{IV} is intended to be checked by the individual voter who instantiates the test with her private information (*e.g.*, her vote and data derived during the execution of the protocol) and the public information available on the bulletin board. Φ^{UV} and Φ^{EV} can be checked by any external observer and only rely on public information, *i.e.*, the contents of the bulletin board.

The consideration of eligibility verifiability is particularly interesting as it provides an assurance that the election outcome corresponds to votes legitimately cast and hence provides a mechanism to detect ballot stuffing. We note that this property has been largely neglected in previous work and an earlier work of ours [SRKK09] only provided limited scope for.

A further interesting aspect of our work is the clear identification of which parts of the voting system need to be trusted to achieve verifiability. All untrusted parts of the system will be controlled by the adversarial environment and do not need to be modelled. Ideally, such a process would only model the interaction between a *voter* and the voting terminal; *that is, the messages input by the voter*. In particular, the voter should not need to trust the election hardware or software. However, achieving absolute verifiability in this context is difficult and one often needs to trust some parts of the voting software or some administrators. Such trust assumptions are motivated by the fact that parts of a protocol can be audited, or can be executed in a distributed manner amongst several different election officials. For instance, in Helios 2.0 [Adi08], the ballot construction can be audited using a cast-or-audit mechanism. Whether trust assumptions are reasonable depends on the context of the given election, but our work makes them explicit.

Of course the tests $\Phi^{\mathsf{IV}}, \Phi^{\mathsf{UV}}$ and Φ^{EV} need to be verified in a trusted environment (if a test is checked by malicious software that always evaluates the test to hold, it is useless). However, the verification of these tests, unlike the election, can be repeated on different machines, using different software, provided by different stakeholders of the election. Another possibility to avoid this issue would be to have tests which are human-verifiable as discussed in [Adi06, Chapter 5].

This section is based on the results presented in [KRS10].

2.1 Formalising voting protocols for verifiability properties

To model verifiability properties we add a *record* construct to the applied pi calculus. We assume an infinite set of distinguished *record variables* $r, r_1, \ldots$. The syntax of plain processes is extended by the construct $\mathsf{rec}(r, M).P$. We write $fn(A)$ and $fn(M)$ for the set of record variables in a process and a term. Intuitively, the record message construct $\mathsf{rec}(r, M).P$ introduces the possibility to enter special entries in frames. We suppose that the sort system ensures that r is a variable of record sort, which may only be used as a first argument of the rec construct or in the domain of the frame. Moreover, we make the global assumption that a record variable has a unique occurrence in each process. Intuitively, this construct will be used to allow a voter to privately record some information which she may later use to verify the election.

As discussed in the introduction we want to explicitly specify the parts of the election protocol which need to be trusted. Formally the trusted parts of the voting protocol can be captured using a voting process specification.

Definition 5 (Voting process specification). *A* voting process specification *is a tuple* $\langle V, A\rangle$ *where* V *is a plain process without replication and* A *is a closed evaluation context such that* $fv(V) = \{v\}$ *and* $fn(V) = \emptyset$.

For the purposes of individual verifiability the voter may rely on some data derived during the protocol execution. We must therefore keep track of all such values, which is achieved using the record construct. Given a finite process P without replication we denote by $\mathsf{R}(P)$, the process which records any freshly generated name and any input, *i.e.*, we replace any occurence of νn with $\nu n.\mathsf{rec}(r, n)$ and $\mathsf{in}(u, x)$ with $\mathsf{in}(u, x).\mathsf{rec}(r, x)$ for some fresh record variable r for each replacement.

Definition 6. *Given a voting process specification* $\langle V, A\rangle$*, integer* $n \in \mathbb{N}$*, and names* $s_1, \ldots, s_n$*, we build the augmented voting process* $\mathsf{VP}_n^+(s_1, \ldots, s_n) = A[V_1^+ \mid \cdots \mid V_n^+]$ *where* $V_i^+ = \mathsf{R}(V)\{^{s_i}/_v\}\{^{r_i}/_r \mid r \in fn(\mathsf{R}(V))\}$.

Given a sequence of record variables $\tilde{r}$, we denote by $\tilde{r}_i$ the sequence of variables obtained by indexing each variable in $\tilde{r}$ with i. The process $\mathsf{VP}_n^+(s_1, \ldots, s_n)$ models the voting protocol for n voters casting votes $s_1, \ldots, s_n$, who privately record the data that may be needed for verification using record variables $\tilde{r}_i$.

2.2 Election verifiability

We formalise election verifiability using three tests Φ^{IV}, Φ^{UV}, Φ^{EV}. Formally, a test is built from conjunctions and disjunctions of *atomic tests* of the form $(M =_E N)$ where M, N are terms. Tests may contain variables and will need to hold on frames arising from arbitrary protocol executions. We now recall the purpose of each test and assume some naming conventions about variables.
Individual verifiability: The test Φ^{IV} allows a voter to identify her ballot in the bulletin board. The test has:

- a variable v referring to a voter's vote.
- a variable w referring to a voter's public credential.
- some variables $x, \bar{x}, \hat{x}, \ldots$ expected to refer to global public values pertaining to the election, *e.g.*, public keys belonging to election administrators.
- a variable y expected to refer to the voter's ballot on the bulletin board.
- some record variables $r_1, \ldots, r_k$ referring to the voter's private data.

Universal verifiability: The test Φ^{UV} allows an observer to check that the election outcome corresponds to the ballots in the bulletin board. The test has:

- a tuple of variables $\tilde{v} = (v_1, \ldots, v_n)$ referring to the declared outcome.
- some variables $x, \bar{x}, \hat{x}, \ldots$ as above.
- a tuple $\tilde{y} = (y_1, \ldots, y_n)$ expected to refer to all the voters' ballots on the bulletin board.
- some variables $z, \bar{z}, \hat{z}, \ldots$ expected to refer to outputs generated during the protocol used for the purposes of universal and eligibility verification.

Eligibility verifiability: The test Φ^{EV} allows an observer to check that each ballot in the bulletin board was cast by a unique registered voter. The test has:

- a tuple $\tilde{w} = (w_1, \ldots, w_n)$ referring to public credentials of eligible voters.
- a tuple $\tilde{y}$, variables $x, \bar{x}, \hat{x}, \ldots$ and variables $z, \bar{z}, \hat{z}, \ldots$ as above.

Individual and universal verifiability. The tests suitable for the purposes of election verifiability have to satisfy certain conditions: if the tests succeed, then the data output by the election is indeed valid (*soundness*); and there is a behaviour of the election authority which produces election data satisfying the tests (*effectiveness*). Formally these requirements are captured by the definition below. We write $\tilde{T} \simeq \tilde{T}'$ to denote that the tuples $\tilde{T}$ and $\tilde{T}'$ are a permutation of each others modulo the equational theory, that is, we have $\tilde{T} = T_1, \ldots T_n$, $\tilde{T}' = T'_1, \ldots T'_n$ and there exists a permutation χ on $\{1, \ldots, n\}$ such that for all $1 \leq i \leq n$ we have $T_i =_E T'_{\chi(i)}$.

Definition 7 (Individual and universal verifiability). *A voting specification $\langle V, A\rangle$ satisfies* individual and universal verifiability *if for all $n \in \mathbb{N}$ there exist tests $\Phi^{\mathsf{IV}}, \Phi^{\mathsf{UV}}$ such that $fn(\Phi^{\mathsf{IV}}) = fn(\Phi^{\mathsf{UV}}) = fn(\Phi^{\mathsf{UV}}) = \emptyset$, $fn(\Phi^{\mathsf{IV}}) \subseteq fn(\mathsf{R}(V))$, and for all names $\tilde{s} = (s_1, \ldots, s_n)$ the conditions below hold. Let $\tilde{r} = fn(\Phi^{\mathsf{IV}})$ and $\Phi^{\mathsf{IV}}_i = \Phi^{\mathsf{IV}}\{^{s_i}/_{v}, ^{\tilde{r}_i}/_{\tilde{r}}\}$.*

Soundness. For all contexts C and processes B such that $C[\mathsf{VP}^+_n(s_1, \ldots, s_n)] \Longrightarrow B$ and $\phi(B) \equiv \nu\tilde{n}.\sigma$, we have:

$$\forall i, j. \quad \Phi^{\mathsf{IV}}_i\sigma \wedge \Phi^{\mathsf{IV}}_j\sigma \Rightarrow i = j \tag{4}$$

$$\Phi^{\mathsf{UV}}\sigma \wedge \Phi^{\mathsf{UV}}\{^{\tilde{v}'}/_{\tilde{v}}\}\sigma \Rightarrow \tilde{v}\sigma \simeq \tilde{v}'\sigma \tag{5}$$

$$\bigwedge_{1 \leq i \leq n} \Phi^{\mathsf{IV}}_i\{^{y_i}/_{y}\}\sigma \wedge \Phi^{\mathsf{UV}}\sigma \;\Rightarrow\; \tilde{s} \simeq \tilde{v}\sigma \tag{6}$$

Effectiveness. There exists a context C and a process B, such that $C[\mathsf{VP}^+_n(s_1, \ldots, s_n)] \Longrightarrow B$, $\phi(B) \equiv \nu\tilde{n}.\sigma$ and

$$\bigwedge_{1 \leq i \leq n} \Phi^{\mathsf{IV}}_i\{^{y_i}/_{y}\}\sigma \wedge \Phi^{\mathsf{UV}}\sigma \tag{7}$$

An individual voter should verify that the test Φ^{IV} holds when instantiated with her vote s_i, the information $\tilde{r}_i$ recorded during the execution of the protocol and some bulletin board entry. Indeed, Condition (4) ensures that the test will hold for at most one bulletin board entry. (Note that Φ^{IV}_i and Φ^{IV}_j are evaluated with the same ballot $y\sigma$ provided by $C[\_]$.) The fact that her ballot is counted will be ensured by Φ^{UV} which should also be tested by the voter. An observer will instantiate the test Φ^{UV} with the bulletin board entries $\tilde{y}$ and the declared outcome $\tilde{v}$. Condition (5) ensures the observer that Φ^{UV} only holds for a single outcome. Condition (6) ensures that if a bulletin board contains the ballots of voters who voted $s_1, \ldots, s_n$ then Φ^{UV} only holds if the declared outcome is (a permutation of) these votes. Finally, Condition (7) ensures that there exists an execution where the tests hold. In particular this allows us to verify whether the protocol can satisfy the tests when executed as expected. This also avoids tests which are always false and would make Conditions (4)–(6) vacuously hold.

Eligibility verifiability. To fully capture *election verifiability*, the tests Φ^{IV} and Φ^{UV} must be supplemented by a test Φ^{EV} that checks eligibility of the voters whose votes have been counted. We suppose that the public credentials of eligible voters appear on the bulletin board. Φ^{EV} allows an observer to check that only these individuals (that is, those in possession of credentials) cast votes, and at most one vote each.

Definition 8 (Election verifiability). *A voting specification $\langle V, A\rangle$ satisfies* election verifiability *if for all $n \in \mathbb{N}$ there exist tests $\Phi^{\mathsf{IV}}, \Phi^{\mathsf{UV}}, \Phi^{\mathsf{EV}}$ such that $fn(\Phi^{\mathsf{IV}}) = fn(\Phi^{\mathsf{UV}}) = fn(\Phi^{\mathsf{EV}}) = fn(\Phi^{\mathsf{UV}}) = fn(\Phi^{\mathsf{EV}}) = \emptyset$, $fn(\Phi^{\mathsf{IV}}) \subseteq fn(\mathsf{R}(V))$, and for all names $\tilde{s} = (s_1, \ldots, s_n)$ we have:*

1. *The tests Φ^{IV} and Φ^{UV} satisfy each of the conditions of Definition 7;*
2. *The additional conditions 8, 9, 10 and 11 below hold.*

Let $\tilde{r} = fn(\Phi^{\mathsf{IV}})$, $\Phi^{\mathsf{IV}}_i = \Phi^{\mathsf{IV}}\{^{s_i}/_{v}, ^{\tilde{r}_i}/_{\tilde{r}}, ^{y_i}/_{y}\}$, $X = fv(\Phi^{\mathsf{EV}}) \backslash \mathsf{dom}\mathsf{VP}^+_{\mathsf{n}}(s_1, \ldots, s_n)$

Soundness. For all contexts C and processes B such that $C[\mathsf{VP}^+_{\mathsf{n}}(s_1,\ldots,s_n)] \Longrightarrow B$ and $\phi(B) \equiv \nu\tilde{n}.\sigma$, we have:

$$\Phi^{\mathsf{EV}}\sigma \wedge \Phi^{\mathsf{EV}}\{^{x'}/_{x} \mid x \in X\backslash\tilde{y}\}\sigma \Rightarrow \tilde{w}\sigma \simeq \tilde{w}'\sigma \tag{8}$$

$$\bigwedge_{1\leq i\leq n} \Phi^{\mathsf{IV}}_i\sigma \wedge \Phi^{\mathsf{EV}}\{^{\tilde{w}'}/_{\tilde{w}}\}\sigma \quad \Rightarrow \quad \tilde{w}\sigma \simeq \tilde{w}'\sigma \tag{9}$$

$$\Phi^{\mathsf{EV}}\sigma \wedge \Phi^{\mathsf{EV}}\{^{x'}/_{x} \mid x \in X\backslash\tilde{w}\}\sigma \Rightarrow \tilde{y}\sigma \simeq \tilde{y}'\sigma \tag{10}$$

Effectiveness. There exists a context C and a process B such that $C[\mathsf{VP}^+_{\mathsf{n}}(s_1,\ldots,s_n)] \Longrightarrow B$, $\phi(B) \equiv \nu\tilde{n}.\sigma$ and

$$\bigwedge_{1\leq i\leq n} \Phi^{\mathsf{IV}}_i\sigma \wedge \Phi^{\mathsf{UV}}\sigma \wedge \Phi^{\mathsf{EV}}\sigma \tag{11}$$

The test Φ^{EV} is instantiated by an observer with the bulletin board. Condition (8) ensures that, given a set of ballots $\tilde{y}\sigma$, provided by the environment, Φ^{EV} succeeds only for one list of voter public credentials. Condition (9) ensures that if a bulletin board contains the ballots of voters with public credentials $\tilde{w}\sigma$ then Φ^{EV} only holds on a permutation of these credentials. Condition (10) ensures that, given a set of credentials $\tilde{w}$, only one set of bulletin board entries $\tilde{y}$ are accepted by Φ^{EV} (observe that for such a strong requirement to hold we expect the voting specification's frame to contain a public key, to root trust). Finally, the effectiveness condition is similar to Condition (7) of Definition 7.

2.3 Case studies

We have analysed verifiability in the protocols by Fujioka et al. [FOO92] and Juels et al. [JCJ05] recently implemented as Civitas [CCM08]. In particular for each of these protocols we identify the exact parts of the system and software that need to be trusted. As an illustration we consider the protocol by Fujioka et al. [FOO92].

Definition 9. *The voting process specification $\langle V_{\mathsf{foo}}, A_{\mathsf{foo}}\rangle$ is defined as*

$$V_{\mathsf{foo}} \mathrel{\hat{=}} \nu rnd.\mathsf{out}v.\mathsf{out}rnd \quad \text{and} \quad A_{\mathsf{foo}}[\_] \mathrel{\hat{=}} \_$$

Intuitively, this specification says that the voter only needs to enter into a terminal a fresh random value rnd and a vote v. The voter does not need to trust any other parts of the system or the administrators. Whether, a voter can generate a fresh random value (which is expected to be used as the key for a commitment), enter it in a terminal and remember it for verifiability or whether some software is trustworthy to achive this task is questionable. Our analysis makes this hypothesis explicit. We have shown that the above voting specification indeed respects individual and universal verifiability.

Theorem 1. *$\langle V_{\mathsf{foo}}, A_{\mathsf{foo}}\rangle$ satisfies individual and universal verifiability.*

However, the protocol by Fujioka et al. does not satisfy eligibility verifiability (even if all the parts of the protocol are trusted).

JCJ/Civitas does achieve full election verifiability considering the following trust assumptions.

- The voter is able to construct her ballot; that is, she is able to generate nonces m, m', construct a pair of ciphertexts and generate a zero-knowledge proof.
- The registrar constructs distinct credentials d for each voter and constructs the voter's public credential correctly. (The latter assumption can be dropped if the registrar provides a proof that the public credential is correctly formed [JCJ05].) The registrar also keeps the private part of the signing key secret.

The analyses were carried out by hand, but the proofs were surprisingly straightforward.

References

[Adi06] Ben Adida. *Advances in Cryptographic Voting Systems*. PhD thesis, MIT, 2006.

[Adi08] Ben Adida. Helios: Web-based open-audit voting. In *Proc. 17th Usenix Security Symposium*, pages 335–348. USENIX Association, 2008.

[BHM08] Michael Backes, Catalin Hritcu, and Matteo Maffei. Automated verification of remote electronic voting protocols in the applied pi-calculus. In *Proc. 21st IEEE Computer Security Foundations Symposium, (CSF'08)*, pages 195–209. IEEE Comp. Soc. Press, 2008.

[BT94] Josh Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *Proc. 26th Symposium on Theory of Computing (STOC'94)*, pages 544–553, Montréal, Québec, 1994. ACM Press.

[CCM08] Michael R. Clarkson, Stephen Chong, and Andrew C. Myers. Civitas: Toward a secure voting system. In *Proc. Symposium on Security and Privacy (SP'08)*, pages 354–368, Washington, DC, USA, 2008. IEEE Computer Society.

[DC] Roberto Di Cosmo. On privacy and anonymity in electronic and non electronic voting: the ballot-as-signature attack.

[DKR09] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, July 2009.

[FOO92] Atsushi Fujioka, Tatsuaki Okamoto, and Kazui Ohta. A practical secret voting scheme for large scale elections. In J. Seberry and Y. Zheng, editors, *Advances in Cryptology – AUSCRYPT '92*, volume 718 of *Lecture Notes in Computer Science*, pages 244–251. Springer, 1992.

[JCJ05] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In *Proc. Workshop on Privacy in the Electronic Society (WPES'05)*, Alexandria, USA, 2005. ACM Press.

[KRS10] Steve Kremer, Mark D. Ryan, and Ben Smyth. Election verifiability in electronic voting protocols. In Dimitris Gritzalis and Bart Preneel, editors, *Proceedings of the 15th European Symposium on Research in Computer Security (ESORICS'10)*, Lecture Notes in Computer Science, Athens, Greece, September 2010. Springer. To appear.

[LBD+04] Byoungcheon Lee, Colin Boyd, Ed Dawson, Kwangjo Kim, Jeongmo Yang, and Seungjae Yoo. Providing receipt-freeness in mixnet-based voting protocols. In Jong In Lim and Dong Hoon Lee, editors, *Proc. Information Security and Cryptology (ICISC'03)*, volume 2971 of *Lecture Notes in Computer Science*, pages 245–258, Seoul, Korea, 2004. Springer.

[MN06] T. Moran and M. Naor. Receipt-free universally-verifiable voting with everlasting privacy. In *Advances in Cryptology - CRYPTO'06*, volume 4117 of *Lecture Notes in Computer Science*, pages 373–392. Springer, 2006.

[Oka96] Tatsuaki Okamoto. An electronic voting scheme. In *Proc. IFIP World Conference on IT Tools*, pages 21–30, Canberra, Australia, 1996.

[Oka97] Tatsuaki Okamoto. Receipt-free electronic voting schemes for large scale elections. In *Proc. 5th Int. Security Protocols Workshop*, volume 1361 of *Lecture Notes in Computer Science*, pages 25–35, Paris, France, 1997. Springer.

[SRKK09] Ben Smyth, Mark D. Ryan, Steve Kremer, and Mounira Kourjieh. Election verifiability in electronic voting protocols (preliminary version). In Olivier Pereira, Jean-Jacques Quisquater, and François-Xavier Standaert, editors, *Proceedings of the 4th Benelux Workshop on Information and System Security (WISSEC'09)*, Louvain-la-Neuve, Belgium, November 2009.

[SS96] Steve Schneider and Abraham Sidiropoulos. CSP and anonymity. In *Proc. 4th European Symposium On Research In Computer Security (ESORICS'96)*, volume 1146 of *Lecture Notes in Computer Science*, pages 198–218. Springer, 1996.

[UM10] Dominique Unruh and Jörn Müller-Quade. Universally composable incoercibility. In *Advances in Cryptology - CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 411–428, 2010.

Verifying privacy-type properties of electronic voting protocols ⋆

Stéphanie Delaune [a,b], Steve Kremer [b], Mark Ryan [a]

[a] *School of Computer Science, University of Birmingham, UK*

[b] *LSV, CNRS & ENS Cachan & INRIA Futurs projet SECSI, France*

Abstract

Electronic voting promises the possibility of a convenient, efficient and secure facility for recording and tallying votes in an election. Recently highlighted inadequacies of implemented systems have demonstrated the importance of formally verifying the underlying voting protocols. We study three privacy-type properties of electronic voting protocols: in increasing order of strength, they are vote-privacy, receipt-freeness, and coercion-resistance.

We use the applied pi calculus, a formalism well adapted to modelling such protocols, which has the advantages of being based on well-understood concepts. The privacy-type properties are expressed using observational equivalence and we show in accordance with intuition that coercion-resistance implies receipt-freeness, which implies vote-privacy.

We illustrate our definitions on three electronic voting protocols from the literature. Ideally, these three properties should hold even if the election officials are corrupt. However, protocols that were designed to satisfy receipt-freeness or coercion-resistance may not do so in the presence of corrupt officials. Our model and definitions allow us to specify and easily change which authorities are supposed to be trustworthy.

Key words: voting protocol, applied pi calculus, formal methods, privacy and anonymity properties.

⋆ This work has been partly supported by the EPSRC projects EP/E029833, *Verifying Properties in Electronic Voting Protocols* and EP/E040829/1, *Verifying anonymity and privacy properties of security protocols*, the ARA SESUR project AVOTÉ and the ARTIST2 NoE.

1 Introduction

Electronic voting protocols. Electronic voting promises the possibility of a convenient, efficient and secure facility for recording and tallying votes. It can be used for a variety of types of elections, from small committees or on-line communities through to full-scale national elections. Electronic voting protocols are formal protocols that specify the messages sent between the voters and administrators. Such protocols have been studied for several decades. They offer the possibility of abstract analysis of the voting system against formally-stated properties.

In this paper, we recall some existing protocols which have been developed over the last decades, and some of the security properties they are intended to satisfy. We focus on privacy-type properties. We present a framework for analysing those protocols and determining whether they satisfy the properties.

From the protocol point of view, the main challenge in designing an election system is to guarantee *vote-privacy*. We may distinguish three main kinds of protocols in the literature, classified according to the mechanism they employ to guarantee privacy. In *blind signature schemes* [15,24,30,35], the voter first obtains a token, which is a message blindly signed by the administrator and known only to the voter herself. The signature of the administrator confirms the voter's eligibility to vote. She later sends her vote anonymously, with this token as proof of eligibility. In schemes using *homomorphic encryption* [6,27], the voter cooperates with the administrator in order to construct an encryption of her vote. The administrator then exploits homomorphic properties of the encryption algorithm to compute the encrypted tally directly from the encrypted votes. A third kind of scheme uses randomisation (for example by mixnets) to mix up the votes so that the link between voter and vote is lost [16,17]. Our focus in this paper is on protocols of the first type, although our methods can probably be used for protocols of the second type. Because it involves mixes, which are probabilistic, the third type is hard to address with our methods that are purely non-deterministic.

Properties of electronic voting protocols. Some properties commonly sought for voting protocols are the following:

- Eligibility: only legitimate voters can vote, and only once.
- Fairness: no early results can be obtained which could influence the remaining voters.
- Individual verifiability: a voter can verify that her vote was really counted.
- Universal verifiability: the published outcome really is the sum of all the votes.
- Vote-privacy: the fact that a particular voter voted in a particular way is not revealed to anyone.

- Receipt-freeness: a voter does not gain any information (a *receipt*) which can be used to prove to a coercer that she voted in a certain way.
- Coercion-resistance: a voter cannot cooperate with a coercer to prove to him that she voted in a certain way.

The last three of these are broadly *privacy-type* properties since they guarantee that the link between the voter and her vote is not revealed by the protocol.

The weakest of the three, called *vote-privacy*, roughly states that the fact that a voter voted in a particular way is not revealed to anyone. When stated in this simple way, however, the property is in general false, because if all the voters vote unanimously then everyone will get to know how everyone else voted. The formalisation we give in this paper in fact says that no party receives information which would allow them to distinguish one situation from another one in which two voters swap their votes.

Receipt-freeness says that the voter does not obtain any artefact (a "receipt") which can be used later to prove to another party how she voted. Such a receipt may be intentional or unintentional on the part of the designer of the system. Unintentional receipts might include nonces or keys which the voter is given during the protocol. Receipt-freeness is a stronger property than privacy. Intuitively, privacy says that an attacker cannot discern how a voter votes from any information that the voter necessarily reveals during the course of the election. Receipt-freeness says the same thing even if the voter voluntarily reveals additional information.

Coercion-resistance is the third and strongest of the three privacy properties. Again, it says that the link between a voter and her vote cannot be established by an attacker, this time even if the voter cooperates with the attacker during the election process. Such cooperation can include giving to the attacker any data which she gets during the voting process, and using data which the attacker provides in return. When analysing coercion-resistance, we assume that the voter and the attacker can communicate and exchange data at any time during the election process. Coercion-resistance is intuitively stronger than receipt-freeness, since the attacker has more capabilities.

Of course, the voter can simply tell an attacker how she voted, but unless she provides convincing evidence the attacker has no reason to believe her. Receipt-freeness and coercion-resistance assert that she cannot provide convincing evidence.

Coercion-resistance cannot possibly hold if the coercer can physically vote on behalf of the voter. Some mechanism is necessary for isolating the voter from the coercer at the moment she casts her vote. This can be realised by a voting booth, which we model here as a private and anonymous channel between the voter and the election administrators.

Note that in literature the distinction between receipt-freeness and coercion-resistance is not very clear. The definitions are usually given in natural language and are insufficiently precise to allow comparison. The notion of receipt-freeness first appeared in the work of Benaloh and Tuinstra [7]. Since then, several schemes [7,39] were proposed in order to meet the condition of receipt-freeness, but later shown not to satisfy it. One of the reasons for such flaws is that no formal definition of receipt-freeness has been given. The situation for coercion-resistance is similar. Systems have been proposed aiming to satisfy it; for example, Okamoto [40] presents a system resistant to interactive coercers, thus aiming to satisfy what we call coercion-resistance, but this property is stated only in natural language. Recently, a rigorous definition in a computational model has been proposed by Juels *et al.* for coercion-resistance [31]. We present in this paper what we believe to be the first "formal methods" definition of receipt-freeness and coercion-resistance. It is difficult to compare our definition and the one proposed by Juels *et al.* [31] due to the inherently different models.

Verifying electronic voting protocols. Because security protocols in general are notoriously difficult to design and analyse, formal verification techniques are particularly important. In several cases, protocols which were thought to be correct for several years have, by means of formal verification techniques, been discovered to have major flaws. Our aim in this paper is to use and develop verification techniques, focusing on the three privacy-type properties mentioned above. We choose *the applied pi calculus* [2] as our basic modelling formalism, which has the advantages of being based on well-understood concepts. The applied pi calculus has a family of proof techniques which we can use, and it is partly supported by the ProVerif tool [8]. Moreover, the applied pi calculus allows us to reason about equational theories in order to model the wide variety of cryptographic primitives often used in voting protocols.

As it is often done in protocol analysis, we assume the Dolev-Yao abstraction: cryptographic primitives are assumed to work perfectly, and the attacker controls the public channels. The attacker can see, intercept and insert messages on public channels, but can only encrypt, decrypt, sign messages or perform other cryptographic operations if he has the relevant key. In general, we assume that the attacker also controls the election officials, since the protocols we investigate are supposed to be resistant even if the officials are corrupt. Some of the protocols explicitly require a trusted device, such as a smart card; we do not assume that the attacker controls those devices.

How the properties are formalised. As already mentioned, the vote-privacy property is formalised as the assertion that the attacker does not receive information which enables him to distinguish a situation from another one in which two voters swap their votes. In other words, the attacker cannot distinguish a situation

in which Alice votes a and Bob votes b, from another one in which they vote the other way around. This is formalised as an observational equivalence property in applied pi.

Receipt-freeness is also formalised as an observational equivalence. Intuitively, a protocol is receipt-free if the attacker cannot detect a difference between Alice voting in the way he instructed, and her voting in some other way, provided Bob votes in the complementary way each time. As in the case of privacy, Bob's vote is required to prevent the observer seeing a different number of votes for each candidate. Alice cooperates with the attacker by sharing secrets, but the attacker cannot interact with Alice to give her some prepared messages.

Coercion-resistance is formalised as an observational equivalence too. In the case of coercion-resistance, the attacker (which we may also call the coercer) is assumed to communicate with Alice during the protocol, and can prepare messages which she should send during the election process. This gives the coercer much more power.

Ideally, these three properties should hold even if the election officials are corrupt. However, protocols that were designed to satisfy vote-privacy, receipt-freeness or coercion-resistance do not necessarily do so in the presence of corrupt officials. Our model and definitions allow us to specify and easily change which authorities are supposed to be trustworthy.

Related properties and formalisations. The idea of formalising privacy-type properties as some kind of observational equivalence in a process algebra or calculus goes back to the work of Schneider and Sidiropoulos [42]. Similar ideas have been used among others by Fournet and Abadi [23], Mauw *et al.* [36] as well as Kremer and Ryan [34]. Other formalizations of anonymity are based on epistemic logics, e.g. [26]. All of these definitions are mainly concerned with *possibilistic* definitions of anonymity. It is also possible to define *probabilistic* anonymity, such as in [41,44,26,11], which gives a more fine-grained characterisation of the level of anonymity which has been achieved. In [20,43,12], information-theoretic measures have been proposed to quantify the degree of anonymity. In this paper we only focus on *possibilistic* flavours of privacy-type properties and assume that channels are anonymous (without studying exactly how these channels are implemented).

Receipt-freeness and coercion-resistance are more subtle than simple privacy. They involve the idea that the voter cannot *prove* how she voted to the attacker. This is a special case of incoercible multi-party computation, which has been explored in the computational security setting [10]. Similarly to their definition, we define incoercibility as the ability to present the coercer with fake data which matches the public transcript as well as the real data. Our definition specialises the setting to electronic voting, and is designed for a Dolev-Yao-like model.

Independently of our work, Jonker and de Vink [28] give a logical characterisation of the notion of receipt in electronic voting processes. Jonker and Pieters [29] also define receipt-freeness in epistemic logic. However, while these formalisms may be appealing to reason about the property, they seem less suited for modelling the protocol and attacker capabilities. These logics are geared to expressing properties rather than operational steps of a protocol. Thus, modelling protocols using epistemic-logic-based approaches is tedious and requires a high degree of expertise. Baskar *et al.* [4] present a promising approach defining an epistemic logic for a protocol language.

The "inability to prove" character of coercion-resistance and receipt-freeness is also shared by the property called *abuse-freeness* in contract-signing protocols. A contract-signing protocol is abuse-free if signer Alice cannot prove to an observer that she is in a position to determine the outcome of the contract. Abuse-freeness has been formalised in a Dolev-Yao-like setting [32] as the ability to provide a message that allows the observer to test whether Alice is in such a position. This notion of test is inspired by static equivalence of the applied pi calculus. However, this notion of test is purely *offline*, which is suitable for abuse-freeness. In our formalization the voter may provide data that allows an active adversary to distinguish two processes which yields a more general notion of receipt (probably too general for abuse-freeness).

To the best of our knowledge, our definitions constitute the first observational equivalence formalisations of the notion of *not being able to prove* in the formal methods approach to security.

Electronic voting in the real world. Governments the world over are trialling and adopting electronic voting systems, and the security aspects have been controversial. For example, the electronic voting machines used in recent US elections have been fraught with security problems. Researchers [33] have analysed the source code of the Diebold machines used in 37 US states. This analysis has produced a catalogue of vulnerabilities and possible attacks. More recent work [21] has produced a security study of the Diebold AccuVote-TS voting machine, including both hardware and software. The results shows that it is vulnerable to very serious attacks. For example, an attacker who gets physical access to a machine or its removable memory card for as little as one minute could install malicious code, which could steal votes undetectably, modifying all records, logs, and counters to be consistent with the fraudulent vote count it creates. They also showed how an attacker could create malicious code that spreads automatically from machine to machine during normal election activities. In another study, a Dutch voting machine was reprogrammed to play chess, rather than count votes, which resulted in the machine being removed from use [25].

These real-world deployments do not rely on the kind of formal protocols studied

in this paper, and therefore our work has no direct bearing on them. The protocols studied here are designed to ensure that vote stealing is cryptographically impossible, and the properties of individual and universal verifiability provide guarantee that voters can verify the outcome of the election themselves. It is hoped that work such as ours in proving the security properties of such protocols will promote their take-up by makers of electronic voting equipment. If deployed, these protocols would—at least to some extent—remove the requirement to trust the hardware and software used by election officials, and even to trust the officials themselves.

This paper. We recall the basic ideas and concepts of the applied pi calculus, in Section 2. Next, in Section 3, we present the framework for formalising voting protocols from the literature, and in Section 4 we show how the three privacy-like properties are formalised. Also in Section 4, we investigate the relationships between the properties and we show that the expected implications hold between them. In Sections 5, 6 and 7 we recall three voting protocols from the literature, and show how they can be formalised in our framework. We analyse which of the properties they satisfy.

Some of the results have been published in two previous papers [34,18]. This paper extends and clarifies our results, provides more examples, better explanations, additional case studies and includes proofs. In particular, our definition of coercion-resistance in this paper is much simpler than our previous definition [18], where we relied on a notion we called *adaptive simulation*. That notion turned out to have some counter-intuitive properties, and we have removed it.

2 The applied pi calculus

The applied pi calculus [2] is a language for describing concurrent processes and their interactions. It is based on the pi calculus, but is intended to be less pure and therefore more convenient to use. The applied pi calculus is, in some sense, similar to the spi calculus [3]. The key difference between the two formalisms concerns the way that cryptographic primitives are handled. The spi calculus has a fixed set of primitives built-in (symmetric and public-key encryption), while the applied pi calculus allows one to define less usual primitives (often used in electronic voting protocols) by means of an equational theory. The applied pi calculus has been used to study a variety of security protocols, such as a private authentication protocol [23] or a key establishment protocol [1].

To describe processes in the applied pi calculus, one starts with a set of *names* (which are used to name communication channels or other atomic data), a set of *variables*, and a *signature* Σ which consists of the *function symbols* which will be used to define *terms*. In the case of security protocols, typical function symbols will include enc for encryption, which takes plaintext and a key and returns the corresponding ciphertext, and dec for decryption, taking ciphertext and a key and returning the plaintext. Terms are defined as names, variables, and function symbols applied to other terms. Terms and function symbols are sorted, and of course function symbol application must respect sorts and arities. By the means of an equational theory E we describe the equations which hold on terms built from the signature. We denote $=_{\mathsf{E}}$ the equivalence relation induced by E. A typical example of an equational theory useful for cryptographic protocols is $\mathsf{dec}(\mathsf{enc}(x, k), k) = x$. In this theory, the terms $T_1 = \mathsf{dec}(\mathsf{enc}(\mathsf{enc}(n, k_1), k_2), k_2)$ and $T_2 = \mathsf{enc}(n, k_1)$ are equal, we have $T_1 =_{\mathsf{E}} T_2$ (while obviously the syntactic equality $T_1 = T_2$ does not hold). Two terms are related by $=_{\mathsf{E}}$ only if that fact can be derived from the equations in E. When the set of variables occurring in a term T is empty, we say that T is *ground*.

In the applied pi calculus, one has *plain processes* and *extended processes*. Plain processes are built up in a similar way to processes in the pi calculus, except that messages can contain terms (rather than just names). In the grammar described below, M and N are terms, n is a name, x a variable and u is a metavariable, standing either for a name or a variable.

$P, Q, R :=$	plain processes
0	null process
$P \mid Q$	parallel composition
$!P$	replication
$\nu n.P$	name restriction
if $M = N$ then P else Q	conditional
$\mathsf{in}(u, x).P$	message input
$\mathsf{out}(u, N).P$	message output

We use the notation $\mathsf{in}(u, =M)$ to test whether the input on u is equal (modulo E) to the term M (if it doesn't, the process blocks). Moreover, we sometimes use tuples of terms, denoted by parentheses, while keeping the equational theory for these tuples implicit.

Extended processes add *active substitutions* and restriction on variables:

$A, B, C :=$	extended processes
P	plain process

$A \mid B$	parallel composition
$\nu n.A$	name restriction
$\nu x.A$	variable restriction
$\{^M/_x\}$	active substitution

$\{^M/_x\}$ is the substitution that replaces the variable x with the term M. Active substitutions generalise "let". The process $\nu x.(\{^M/_x\} \mid P)$ corresponds exactly to the process "let $x = M$ in P". As usual, names and variables have scopes, which are delimited by restrictions and by inputs. We write $fv(A)$, $bv(A)$, $fn(A)$ and $bn(A)$ for the sets of free and bound variables and free and bound names of A, respectively. We also assume that, in an extended process, there is at most one substitution for each variable, and there is exactly one when the variable is restricted. We say that an extended process is *closed* if all its variables are either bound or defined by an active substitution.

Active substitutions are useful because they allow us to map an extended process A to its *frame* $\phi(A)$ by replacing every plain process in A with 0. A frame is an extended process built up from 0 and active substitutions by parallel composition and restriction. The frame $\phi(A)$ can be viewed as an approximation of A that accounts for the static knowledge A exposes to its environment, but not A's dynamic behaviour.

Example 1 *For instance, consider the extended processes $A_1 = \{^{M_1}/_{x_1}\} \mid \{^{M_2}/_{x_2}\} \mid P_1$ and $A_2 = \{^{M_1}/_{x_1}\} \mid \{^{M_2}/_{x_2}\} \mid P_2$. Even if these two processes are different from the point of view of their dynamic behaviour, the frames $\phi(A_1)$ and $\phi(A_2)$ are equal. This witnesses the fact that A_1 and A_2 have the same static knowledge.*

The domain of a frame φ, denoted by $\mathrm{dom}(\varphi)$, is the set of variables for which φ defines a substitution (those variables x for which φ contains a substitution $\{^M/_x\}$ not under a restriction on x).

An *evaluation context* $C[\_]$ is an extended process with a hole instead of an extended process. Structural equivalence, noted $\equiv$, is the smallest equivalence relation on extended processes that is closed under α-conversion on names and variables, by application of evaluation contexts, and such that

PAR-0	$A \mid 0 \equiv A$	REPL	$!P \equiv P \mid !P$
PAR-A	$A \mid (B \mid C) \equiv (A \mid B) \mid C$	REWRITE	$\{^M/_x\} \equiv \{^N/_x\}$
PAR-C	$A \mid B \equiv B \mid A$		if $M =_{\mathsf{E}} N$
NEW-0	$\nu n.0 \equiv 0$	ALIAS	$\nu x.\{^M/_x\} \equiv 0$
NEW-C	$\nu u.\nu v.A \equiv \nu v.\nu u.A$	SUBST	$\{^M/_x\} \mid A \equiv \{^M/_x\} \mid A\{^M/_x\}$

NEW-PAR $A \mid \nu u.B \equiv \nu u.(A \mid B)$ if $u \notin fn(A) \cup fv(A)$

Example 2 *Consider the following process P:*

$$\nu s.\nu k.(\mathsf{out}(c_1, \mathsf{enc}(s,k)) \mid \mathsf{in}(c_1, y).\mathsf{out}(c_2, \mathsf{dec}(y,k))).$$

The first component publishes the message $\mathsf{enc}(s,k)$ by sending it on c_1. The second receives a message on c_1, uses the secret key k to decrypt it, and forwards the resulting plaintext on c_2. The process P is structurally equivalent to the following extended process A:

$$A = \nu s, k, x_1.\Big(\mathsf{out}(c_1, x_1) \mid \mathsf{in}(c_1, y).\mathsf{out}(c_2, \mathsf{dec}(y,k)) \mid \{^{\mathsf{enc}(s,k)}/_{x_1}\}\Big)$$

We have $\phi(A) = \nu s, k, x_1.\{^{enc(s,k)}/_{x_1}\} \equiv 0$ (since x_1 is under a restriction).

The following lemma will be useful in the remainder of the paper.

Lemma 3 *Let $C_1 = \nu\tilde{u}_1.(\_ \mid B_1)$ and $C_2 = \nu\tilde{u}_2.(\_ \mid B_2)$ be two evaluation contexts such that $\tilde{u}_1 \cap (fv(B_2) \cup fn(B_2)) = \emptyset$ and $\tilde{u}_2 \cap (fv(B_1) \cup fn(B_1)) = \emptyset$. We have that $C_1[C_2[A]] \equiv C_2[C_1[A]]$ for any extended process A.*

PROOF. Let A be an extended process. We have that

$$\begin{aligned} C_1[C_2[A]] &\equiv \nu\tilde{u}_1.(\nu\tilde{u}_2.(A \mid B_2) \mid B_1) \\ &\equiv \nu\tilde{u}_2.\nu\tilde{u}_1.((A \mid B_1) \mid B_2) \qquad \text{since } \tilde{u}_2 \notin fv(B_1) \cup fn(B_1) \\ &\equiv \nu\tilde{u}_2.(\nu\tilde{u}_1.(A \mid B_1) \mid B_2) \qquad \text{since } \tilde{u}_1 \notin fv(B_2) \cup fn(B_2) \\ &\equiv C_2[C_1[A]] \end{aligned}$$ □

2.2 Semantics

The operational semantics of processes in the applied pi calculus is defined by structural rules defining two relations: *structural equivalence* (briefly described in Section 2.1) and *internal reduction*, noted $\rightarrow$. Internal reduction $\rightarrow$ is the smallest relation on extended processes closed under structural equivalence and application of evaluation contexts such that

(COMM) $\quad \mathsf{out}(a,x).P \mid \mathsf{in}(a,x).Q \rightarrow P \mid Q$

(THEN) $\quad$ if $M = M$ then P else $Q \rightarrow P$

(ELSE) $\quad$ if $M = N$ then P else $Q \rightarrow Q$

for any ground terms M and N such that $M \neq_{\mathsf{E}} N$.

The operational semantics is extended by a *labelled* operational semantics enabling

us to reason about processes that interact with their environment. Labelled operational semantics defines the relation $\xrightarrow{\alpha}$ where α is either an input, or the output of a channel name or a variable of base type.

(IN) $$\mathsf{in}(a,x).P \xrightarrow{in(a,M)} P\{^{M}/_{x}\}$$

(OUT-ATOM) $$\mathsf{out}(a,u).P \xrightarrow{out(a,u)} P$$

(OPEN-ATOM) $$\frac{A \xrightarrow{out(a,u)} A' \qquad u \neq a}{\nu u.A \xrightarrow{\nu u.out(a,u)} A'}$$

(SCOPE) $$\frac{A \xrightarrow{\alpha} A' \qquad u \text{ does not occur in } \alpha}{\nu u.A \xrightarrow{\alpha} \nu u.A'}$$

(PAR) $$\frac{A \xrightarrow{\alpha} A' \quad bv(\alpha) \cap fv(B) = bn(\alpha) \cap fn(B) = \emptyset}{A \mid B \xrightarrow{\alpha} A' \mid B}$$

(STRUCT) $$\frac{A \equiv B \quad B \xrightarrow{\alpha} B' \quad A' \equiv B'}{A \xrightarrow{\alpha} A'}$$

Note that the labelled transition is not closed under application of evaluation contexts. Moreover the output of a term M needs to be made "by reference" using a restricted variable and an active substitution.

Example 4 *Consider the process P defined in Example 2. We have*

$$\begin{array}{rl}
P & \equiv \quad \nu s,k,x_1.(\mathsf{out}(c_1,x_1) \mid \mathsf{in}(c_1,y).\mathsf{out}(c_2,\mathsf{dec}(y,k)) \mid \{^{\mathsf{enc}(s,k)}/_{x_1}\}) \\
\xrightarrow{\nu x_1.out(c_1,x_1)} & \nu s,k.(\mathsf{in}(c_1,y).\mathsf{out}(c_2,\mathsf{dec}(y,k)) \mid \{^{\mathsf{enc}(s,k)}/_{x_1}\}) \\
\xrightarrow{in(c_1,x_1)} & \nu s,k.(\mathsf{out}(c_2,\mathsf{dec}(x_1,k)) \mid \{^{\mathsf{enc}(s,k)}/_{x_1}\}) \\
\equiv & \nu s,k,x_2.(\mathsf{out}(c_2,x_2) \mid \{^{\mathsf{enc}(s,k)}/_{x_1}\} \mid \{^{\mathsf{dec}(x_1,k)}/_{x_2}\}) \\
\xrightarrow{\nu x_2.out(c_1,x_2)} & \nu s,k.(\{^{\mathsf{enc}(s,k)}/_{x_1}\} \mid \{^{\mathsf{dec}(x_1,k)}/_{x_2}\})
\end{array}$$

Let A be the extended process obtained after this sequence of reduction steps. We have that $\phi(A) \equiv \nu s.\nu k.\{^{enc(s,k)}/_{x_1},\ {}^{s}/_{x_2}\}$.

2.3 Equivalences

We can now define what it means for two frames to be *statically equivalent* [2].

Definition 5 (Static equivalence ($\approx_s$)) *Two terms M and N are* equal in the frame ϕ*, written $(M =_{\mathsf{E}} N)\phi$, if, and only if there exists $\tilde{n}$ and a substitution σ such that $\phi \equiv \nu\tilde{n}.\sigma$, $M\sigma =_{\mathsf{E}} N\sigma$, and $\tilde{n} \cap (fn(M) \cup fn(N)) = \emptyset$.*

Two frames ϕ_1 and ϕ_2 are statically equivalent, $\phi_1 \approx_{\mathsf{E}} \phi_2$*, when:*

- $\mathrm{dom}(\phi_1) = \mathrm{dom}(\phi_2)$*, and*
- *for all terms M, N we have that $(M =_{\mathsf{E}} N)\phi_1$ if and only if $(M =_{\mathsf{E}} N)\phi_2$.*

Two extended processes A and B are said to be statically equivalent, denoted by $A \approx_s B$, if we have that $\phi(A) \approx_s \phi(B)$.

Example 6 *Let $\varphi_0 = \nu k.\sigma_0$ and $\varphi_1 = \nu k.\sigma_1$ where $\sigma_0 = \{^{\mathsf{enc}(s_0,k)}/_{x_1}, ^{k}/_{x_2}\}$, $\sigma_1 = \{^{\mathsf{enc}(s_1,k)}/_{x_1}, ^{k}/_{x_2}\}$ and s_0, s_1 and k are names. Let E be the theory defined by the axiom $\mathsf{dec}(\mathsf{enc}(x,k),k) = x$. We have $\mathsf{dec}(x_1,x_2)\sigma_0 =_{\mathsf{E}} s_0$ but not $\mathsf{dec}(x_1,x_2)\sigma_1 =_{\mathsf{E}} s_0$. Therefore we have $\varphi_0 \not\approx_s \varphi_1$. However, note that we have $\nu k.\{^{\mathsf{enc}(s_0,k)}/_{x_1}\} \approx_s \nu k.\{^{\mathsf{enc}(s_1,k)}/_{x_1}\}$.*

Definition 7 (Labelled bisimilarity ($\approx_\ell$)) Labelled bisimilarity *is the largest symmetric relation $\mathcal{R}$ on closed extended processes, such that $A\,\mathcal{R}\,B$ implies*

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

The definition of labelled bisimilarity is like the usual definition of bisimilarity, except that at each step one additionally requires that the processes are statically equivalent. It has been shown that labelled bisimilarity coincides with observational equivalence [2]. We prefer to work with labelled bisimilarity, rather than observational equivalence, because proofs for labelled bisimilarity are generally easier. Labelled bisimilarity can be used to formalise many security properties, in particular anonymity properties, such as those studied in this paper.

When we model protocols in applied pi calculus, we model the honest parties as processes. The dishonest parties are considered to be under the control of the attacker, and are not modelled explicitly. The attacker (together with any parties it controls) form the environment in which the honest processes run. This arrangement implies that we consider only one attacker; to put in another way, we consider that all dishonest parties and attackers share information and trust each other, thus forming a single coalition. This arrangement does not allow us to consider attackers that do not share information with each other.

3 Formalising voting protocols

Before formalising security properties, we need to define what is an electronic voting protocol in applied pi calculus. Different voting protocols often have substantial differences. However, we believe that a large class of voting protocols can be represented by processes corresponding to the following structure.

Definition 8 (Voting process) *A voting process is a closed plain process*

$$VP \equiv \nu\tilde{n}.(V\sigma_1 \mid \cdots \mid V\sigma_n \mid A_1 \mid \cdots \mid A_m).$$

The $V\sigma_i$ are the voter processes, the A_js the election authorities which are required to be honest and the $\tilde{n}$ are channel names. We also suppose that $v \in \mathrm{dom}(\sigma_i)$ is a variable which refers to the value of the vote. We define an evaluation context S which is as VP, but has a hole instead of two of the $V\sigma_i$.

In order to prove a given property, we may require some of the authorities to be honest, while other authorities may be assumed to be corrupted by the attacker. The processes $A_1, \ldots, A_m$ represent the authorities which are required to be honest. The authorities under control of the attacker need not be modelled, since we consider any possible behaviour for the attacker (and therefore any possible behaviour for corrupt authorities). In this case the communications channels are available to the environment.

We have chosen to illustrate our definition with three classical electronic voting protocols of the literature: a protocol due to Fujioka *et al.* [24], a protocol due to Okamoto [39] and one due to Lee *et al.* [35]. After a brief and informal description of those protocols, we formalise them in the applied pi calculus framework in Sections 5, 6 and 7.

4 Formalising privacy-type properties

In this section, we show how the anonymity properties, informally described in the introduction, can be formalised in our setting and we show, in accordance with intuition, that coercion-resistance implies receipt-freeness, which implies privacy. It is rather classical to formalise anonymity properties as some kind of observational equivalence in a process algebra or calculus, going back to the work of Schneider and Sidiropoulos [42]. However, the definition of anonymity properties in the context of voting protocols is rather subtle.

The privacy property aims to guarantee that the link between a given voter V and his vote v remains hidden. Anonymity and privacy properties have been successfully studied using equivalences. However, the definition of privacy in the context of voting protocols is rather subtle. While generally most security properties should hold against an arbitrary number of dishonest participants, arbitrary coalitions do not make sense here. Consider for instance the case where all but one voter are dishonest: as the results of the vote are published at the end, the dishonest voter can collude and determine the vote of the honest voter. A classical trick for modelling anonymity is to ask whether two processes, one in which V_A votes and one in which V_B votes, are equivalent. However, such an equivalence does not hold here as the voters' identities are revealed (and they need to be revealed at least to the administrator to verify eligibility). In a similar way, an equivalence of two processes where only the vote is changed does not hold, because the votes are published at the end of the protocol. To ensure privacy we need to hide the *link* between the voter and the vote and not the voter or the vote itself.

In order to give a reasonable definition of privacy, we need to suppose that at least two voters are honest. We denote the voters V_A and V_B and their votes a, respectively b. We say that a voting protocol respects privacy whenever a process where V_A votes a and V_B votes b is observationally equivalent to a process where V_A votes b and V_B votes a. Formally, privacy is defined as follows.

Definition 9 (Vote-privacy) *A voting protocol respects* vote-privacy *(or just* privacy*) if*

$$S[V_A\{^a/_v\} \mid V_B\{^b/_v\}] \approx_\ell S[V_A\{^b/_v\} \mid V_B\{^a/_v\}]$$

for all possible votes a and b.

The intuition is that if an intruder cannot detect if arbitrary honest voters V_A and V_B swap their votes, then in general he cannot know anything about how V_A (or V_B) voted. Note that this definition is robust even in situations where the result of the election is such that the votes of V_A and V_B are necessarily revealed. For example, if the vote is unanimous, or if all other voters reveal how they voted and thus allow the votes of V_A and V_B to be deduced.

A protocol satisfying privacy also allows arbitrary permutations of votes between voters. For example, we may prove that

$$S[V_A\{^a/_v\} \mid V_B\{^b/_v\} \mid V_C\{^c/_v\}] \approx_\ell S[V_A\{^b/_v\} \mid V_B\{^c/_v\} \mid V_C\{^a/_v\}]$$

as follows:

$$
\begin{aligned}
&S[V_A\{^a/_v\} \mid V_B\{^b/_v\} \mid V_C\{^c/_v\}] \\
&\quad \approx_\ell S[V_A\{^b/_v\} \mid V_B\{^a/_v\} \mid V_C\{^c/_v\}] \text{ using privacy, with } S' = S[\_ \mid V_c\{^c/_v\}] \\
&\quad \approx_\ell S[V_A\{^b/_v\} \mid V_B\{^c/_v\} \mid V_C\{^a/_v\}] \text{ using privacy, with } S'' = S[V_a\{^b/_v\} \mid \_]
\end{aligned}
$$

As already noted, in some protocols the vote-privacy property may hold even if authorities are corrupt, while other protocols may require the authorities to be honest. When proving privacy, we choose which authorities we want to model as honest, by including them in Definition 8 of VP (and hence S).

4.2 Receipt-Freeness

Similarly to privacy, receipt-freeness may be formalised as an observational equivalence. We also formalise receipt-freeness using observational equivalence. However, we need to model the fact that V_A is willing to provide secret information, i.e., the receipt, to the coercer. We assume that the coercer is in fact the attacker who, as usual in the Dolev-Yao model, controls the public channels. To model V_A's communication with the coercer, we consider that V_A executes a voting process which has been modified: any input of base type and any freshly generated names of base type are forwarded to the coercer. We do not forward restricted channel names, as these are used for modelling purposes, such as physically secure channels, e.g. the voting booth, or the existence of a PKI which securely distributes keys (the keys themself are forwarded but not the secret channel name on which the keys are received).

Definition 10 (Process P^{ch}) *Let P be a plain process and ch a channel name. We define P^{ch} as follows:*

- $0^{ch} \mathrel{\widehat{=}} 0$,
- $(P \mid Q)^{ch} \mathrel{\widehat{=}} P^{ch} \mid Q^{ch}$,
- $(\nu n.P)^{ch} \mathrel{\widehat{=}} \nu n.\mathsf{out}(ch, n).P^{ch}$ *when n is name of base type,*
- $(\nu n.P)^{ch} \mathrel{\widehat{=}} \nu n.P^{ch}$ *otherwise,*
- $(\mathsf{in}(u, x).P)^{ch} \mathrel{\widehat{=}} \mathsf{in}(u, x).\mathsf{out}(ch, x).P^{ch}$ *when x is a variable of base type,*
- $(\mathsf{in}(u, x).P)^{ch} \mathrel{\widehat{=}} \mathsf{in}(u, x).P^{ch}$ *otherwise,*
- $(\mathsf{out}(u, M).P)^{ch} \mathrel{\widehat{=}} \mathsf{out}(u, M).P^{ch}$,
- $(!P)^{ch} \mathrel{\widehat{=}} !P^{ch}$,
- *(if $M = N$ then P else Q)^{ch} $\mathrel{\widehat{=}}$ if $M = N$ then P^{ch} else Q^{ch}.*

In the remainder, we assume that $ch \notin fn(P) \cup bn(P)$ before applying the transformation.

Given an extended process A and a channel name ch, we need to define the extended process $A^{\setminus out(ch,\cdot)}$. Intuitively, such a process is as the process A, but hiding the outputs on the channel ch.

Definition 11 (Process $A^{\setminus out(ch,\cdot)}$) *Let A be an extended process. We define the process $A^{\setminus out(ch,\cdot)}$ as $\nu ch.(A \mid !\mathsf{in}(ch, x))$.*

We are now ready to define receipt-freeness. Intuitively, a protocol is receipt-free if, for all voters V_A, the process in which V_A votes according to the intruder's wishes is indistinguishable from the one in which she votes something else. As in the case of privacy, we express this as an observational equivalence to a process in which V_A swaps her vote with V_B, in order to avoid the case in which the intruder can distinguish the situations merely by counting the votes at the end. Suppose the coercer's desired vote is c. Then we define receipt-freeness as follows.

Definition 12 (Receipt-freeness) *A voting protocol is* receipt-free *if there exists a closed plain process V' such that*

- $V'^{\setminus out(chc,\cdot)} \approx_\ell V_A\{^a/_v\}$,
- $S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}] \approx_\ell S[V' \mid V_B\{^c/_v\}]$,

for all possible votes a and c.

As before, the context S in the second equivalence includes those authorities that are assumed to be honest. V' is a process in which voter V_A votes a but communicates with the coercer C in order to feign cooperation with him. Thus, the second equivalence says that the coercer cannot tell the difference between a situation in which V_A genuinely cooperates with him in order to cast the vote c and one in which she pretends to cooperate but actually casts the vote a, provided there is some counterbalancing voter that votes the other way around. The first equivalence of the definition says that if one ignores the outputs V' makes on the coercer channel chc, then V' looks like a voter process V_A voting a.

The first equivalence of the definition may be considered too strong; informally, one might consider that the equivalence should be required only in a particular S context rather than requiring it in any context (with access to all the private channels of the protocol). This would result in a weaker definition, although one which is more difficult to work with. In fact, the variant definition would be only slightly weaker; it is hard to construct a natural example which distinguishes the two possibilities, and in particular it makes no difference to the case studies of later sections. Therefore, we prefer to stick to Definition 12.

According to intuition, if a protocol is receipt-free (for a given set of honest authorities), then it also respects privacy (for the same set):

Proposition 13 *If a voting protocol is receipt-free then it also respects privacy.*

Before we prove this proposition we need to introduce a lemma.

Lemma 14 *Let P be a closed plain process and ch a channel name such that $ch \notin fn(P) \cup bn(P)$. We have $(P^{ch})^{\setminus out(ch,\cdot)} \approx_\ell P$.*

PROOF. (sketch, see Appendix A for details)
We show by induction on the size of P that for any channel name ch such that $ch \notin fn(P) \cup bn(P)$, the equivalence $P^{ch\setminus out(ch,\cdot)} \approx_\ell P$ holds. The base case where $P = 0$ is trivial. Then, we consider the different possibilities for building P. □

PROOF. (of Proposition 13)
By hypothesis, there exists a closed plain process V', such that

- $V'^{\setminus out(chc,\cdot)} \approx_\ell V_A\{^a/_v\}$, and
- $S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}] \approx_\ell S[V' \mid V_B\{^c/_v\}]$.

By applying the evaluation context $\nu chc.(\_ \mid !\mathsf{in}(chc, x))$ on both sides we obtain

$$S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}]^{\setminus out(chc,\cdot)} \approx_\ell S[V' \mid V_B\{^c/_v\}]^{\setminus out(chc,\cdot)}.$$

By using Lemma 3, we obtain that:

- $S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}]^{\setminus out(chc,\cdot)} \equiv S[(V_A\{^c/_v\}^{chc})^{\setminus out(chc,\cdot)} \mid V_B\{^a/_v\}]$,
- $S[V' \mid V_B\{^c/_v\}]^{\setminus out(chc,\cdot)} \equiv S[V'^{\setminus out(chc,\cdot)} \mid V_B\{^c/_v\}]$.

Lastly, thanks to Lemma 14 and the fact that labelled bisimilarity is closed under structural equivalence, we deduce that

$$S[V_A\{^c/_v\} \mid V_B\{^a/_v\}] \approx_\ell S[V'^{\setminus out(chc,\cdot)} \mid V_B\{^c/_v\}].$$

Since we have $V'^{\setminus out(chc,\cdot)} \approx_\ell V_A\{^a/_v\}$, we easily conclude. □

4.3 Coercion-Resistance

Coercion-resistance is a stronger property as we give the coercer the ability to communicate *interactively* with the voter and not only receive information. In this model, the coercer can prepare the messages he wants the voter to send. As for receipt-freeness, we modify the voter process. In the case of coercion-resistance, we give the coercer the possibility to provide the messages the voter should send. The coercer can also decide how the voter branches on *if*-statements.

Definition 15 (Process P^{c_1,c_2}) *Let P be a plain process and c_1, c_2 be channel names. We define P^{c_1,c_2} inductively as follows:*

- $0^{c_1,c_2} \mathrel{\widehat{=}} 0$,
- $(P \mid Q)^{c_1,c_2} \mathrel{\widehat{=}} P^{c_1,c_2} \mid Q^{c_1,c_2}$,
- $(\nu n.P)^{c_1,c_2} \mathrel{\widehat{=}} \nu n.\mathsf{out}(c_1, n).P^{c_1,c_2}$ *when* n *is a name of base type,*
- $(\nu n.P)^{c_1,c_2} \mathrel{\widehat{=}} \nu n.P^{c_1,c_2}$ *otherwise,*
- $(\mathsf{in}(u, x).P)^{c_1,c_2} \mathrel{\widehat{=}} \mathsf{in}(u, x).\mathsf{out}(c_1, x).P^{c_1,c_2}$ *when* x *is a variable of base type,*
- $(\mathsf{in}(u, x).P)^{c_1,c_2} \mathrel{\widehat{=}} \mathsf{in}(u, x).P^{c_1,c_2}$ *otherwise,*
- $(\mathsf{out}(u, M).P)^{c_1,c_2} \mathrel{\widehat{=}} \mathsf{in}(c_2, x).\mathsf{out}(u, x).P^{c_1,c_2}$ *when* M *is a term of base type and* x *is a fresh variable,*
- $(\mathsf{out}(u, M).P)^{c_1,c_2} \mathrel{\widehat{=}} \mathsf{out}(u, M).P^{c_1,c_2}$ *otherwise,*
- $(!P)^{c_1,c_2} \mathrel{\widehat{=}} !P^{c_1,c_2}$,
- *(if* $M = N$ *then* P *else* Q*)*$^{c_1,c_2} \mathrel{\widehat{=}} \mathsf{in}(c_2, x)$. *if* x *= true then* P^{c_1,c_2} *else* Q^{c_1,c_2} *where* x *is a fresh variable and true is a constant.*

As a first approximation, we could try to define coercion-resistance in the following way: a protocol is coercion-resistant if there is a V' such that

$$S[V_A\{^?/_v\}^{c_1,c_2} \mid V_B\{^a/_v\}] \approx_\ell S[V' \mid V_B\{^c/_v\}]. \tag{1}$$

On the left, we have the coerced voter $V_A\{^?/_v\}^{c_1,c_2}$; no matter what she intends to vote (the "?"), the idea is that the coercer will force her to vote c. On the right, the process V' resists coercion, and manages to vote a. Unfortunately, this characterisation has the problem that the coercer could oblige $V_A\{^?/_v\}^{c_1,c_2}$ to vote $c' \neq c$. In that case, the process $V_B\{^c/_v\}$ would not counterbalance the outcome to avoid a trivial way of distinguishing the two sides.

To enable us to reason about the coercer's choice of vote, we model the coercer's behaviour as a context C that defines the interface c_1, c_2 for the voting process. The context C coerces a voter to vote c. Thus, we can characterise coercion-resistance as follows: a protocol is coercion-resistant if there is a V' such that

$$S[C[V_A\{^?/_v\}^{c_1,c_2}] \mid V_B\{^a/_v\}] \approx_\ell S[C[V'] \mid V_B\{^c/_v\}], \tag{2}$$

where C is a context ensuring that the coerced voter $V_A\{^?/_v\}^{c_1,c_2}$ votes c. The context C models the coercer's behaviour, while the environment models the coercer's powers to observe whether the coerced voter behaves as instructed. We additionally require that the context C does not directly use the channel names $\tilde{n}$ restricted by S. Formally one can ensure that $V_A\{^?/_v\}^{c_1,c_2}$ votes c by requiring that $C[V_A\{^?/_v\}^{c_1,c_2}] \approx_\ell V_A\{^c/_v\}^{chc}$. We actually require a slightly weaker condition, $S[C[V_A\{^?/_v\}^{c1,c2}] \mid V_B\{^a/_v\}] \approx_\ell S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}]$, which results in a stronger property.

Putting these ideas together, we arrive at the following definition:

Definition 16 (Coercion-resistance) *A voting protocol is* coercion-resistant *if there exists a closed plain process* V' *such that for any* $C = \nu c_1.\nu c_2.(\_ \mid P)$ *satisfying* $\tilde{n} \cap fn(C) = \emptyset$ *and* $S[C[V_A\{^?/_v\}^{c1,c2}] \mid V_B\{^a/_v\}] \approx_\ell S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}]$*, we*

have

- $C[V']^{\setminus out(chc,\cdot)} \approx_\ell V_A\{^a/_v\}$,
- $S[C[V_A\{^?/_v\}^{c_1,c_2}] \mid V_B\{^a/_v\}] \approx_\ell S[C[V'] \mid V_B\{^c/_v\}]$.

Note that $V_A\{^?/_v\}^{c_1,c_2}$ does not depend on what we put for "?".

The condition that $S[C[V_A\{^?/_v\}^{c1,c2}] \mid V_B\{^a/_v\}] \approx_\ell S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}]$ means that the context C outputs the secrets generated during its computation; this is required so that the environment can make distinctions on the basis of those secrets, as in receipt-freeness. The first bullet point expresses that V' is a voting process for A which fakes the inputs/outputs with C and succeeds in voting a in spite of the coercer. The second bullet point says that the coercer cannot distinguish between V' and the really coerced voter, provided another voter V_B counterbalances.

As in the case of receipt-freeness, the first equivalence of the definition could be made weaker by requiring it only in a particular S context. But we chose not to adopt this extra complication, for the same reasons as given in the case of receipt-freeness.

Remark 17 *The context C models the coercer's behaviour; we can see its role in equivalence (2) as imposing a restriction on the distinguishing power of the environment in equivalence (1). Since the coercer's behaviour is modelled by C while its distinguishing powers are modelled by the environment, it would be useful to write (2) as*

$$C[S[V_A\{^?/_v\}^{c_1,c_2}] \mid V_B\{^a/_v\}]] \approx_\ell C[S[V' \mid V_B\{^c/_v\}]]. \tag{3}$$

Equivalences (2) and (3) are the same (Lemma 3).

According to intuition, if a protocol is coercion-resistant then it respects receipt-freeness too (as before, we keep constant the set of honest authorities):

Proposition 18 *If a voting protocol is coercion-resistant then it also respects receipt-freeness.*

PROOF. Let C be an evaluation context such that $C = \nu c_1.\nu c_2.(\_ \mid P)$ for some plain process P and $S[C[V_A\{^?/_v\}^{c1,c2}] \mid V_B\{^a/_v\}] \approx_\ell S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}]$. Note that such a C can be constructed directly from the vote process V. By hypothesis, we know that there exists a closed plain process V' such that

- $C[V']^{\setminus out(chc,\cdot)} \approx_\ell V_A\{^a/_v\}$,
- $S[C[V_A\{^?/_v\}^{c_1,c_2}] \mid V_B\{^a/_v\}] \approx_\ell S[C[V'] \mid V_B\{^c/_v\}]$.

We need to show that there exists V'' such that

- $V''\backslash_{out(chc,.)} \approx_\ell V_A\{^a/_v\}$,
- $S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}] \approx_\ell S[V'' \mid V_B\{^c/_v\}]$.

Let $V'' = C[V']$. We directly obtain the first requirement. For the second one, we take the hypotheses

- $S[C[V_A\{^?/_v\}^{c_1,c_2}] \mid V_B\{^a/_v\}] \approx_\ell S[C[V'] \mid V_B\{^c/_v\}]$, and
- $S[C[V_A\{^?/_v\}^{c1,c2}] \mid V_B\{^a/_v\}] \approx_\ell S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}]$.

By transitivity of $\approx_\ell$, we obtain $S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}] \approx_\ell S[C[V'] \mid V_B\{^c/_v\}]$. Lastly, we replace $C[V']$ on the right by V''. □

Using the definition of coercion-resistance. To show that a voting protocol satisfies coercion-resistance, it is necessary to give a process V', and it is necessary to show the two bullet points in the definition for all contexts C which satisfy the requirement stated in the definition. In case studies, it is difficult to reason about all possible contexts C, and our analysis is rather informal. In future work, we hope to provide better methods for doing that.

To show that a voting protocol does not satisfy coercion-resistance, it is necessary to show that for all V', there exists a context C for which the bullet points fail. In practice, one may try to give a single C which works for all V'. Since this is a stronger condition, it is sufficient.

5 Protocol due to Fujioka, Okamoto and Ohta

In this section we study a protocol due to Fujioka, Okamoto and Ohta [24]. We first give an informal description of the protocol (see Section 5.1). Then, we show in Section 5.2 how this protocol can be modelled in the applied pi calculus. Lastly, in Section 5.3, we show that the protocol respects privacy. However, the protocol is not receipt-free [39]. The Fujioka, Okamoto and Ohta protocol was also analysed by Nielsen *et al.* [38], but their focus is on properties such as verifiability, eligibility, and fairness, rather than the privacy-type properties of this paper.

5.1 Description

The protocol involves voters, an administrator, verifying that only eligible voters can cast votes, and a collector, collecting and publishing the votes. In comparison with authentication protocols, the protocol also uses some unusual cryptographic primitives such as secure bit-commitment and blind signatures. Moreover, it relies

on anonymous channels. We deliberately do not specify the way these channels are handled as any anonymiser mechanism could be suitable depending on the precise context the protocol is used in. One can use MIX-nets introduced by Chaum [13] whose main idea is to permute and modify (by using decryption or re-encryption) some sequence of objects in order to hide the correspondence between elements of the original and the final sequences. Some other implementations may also be possible, e.g. onion routing [45].

A bit-commitment scheme allows an agent A to commit a value v to another agent B without revealing it immediately. Moreover, B is ensured that A cannot change her mind afterwards and that the value she later reveals will be the same as she thinks at the beginning. For this, A encrypts the value v in some way and sends the encryption to B. The agent B is not able to recover v until A sends him the key.

A blind signature scheme allows a requester to obtain a signature of a message m without revealing the message m to anyone, including the signer. Hence, the signer is requested to sign a message blindly without knowing what he signs. This mechanism is very useful in electronic voting protocol. It allows the voter to obtain a signature of her vote by an authority who checks that she has right to vote without revealing it to the authority.

In a first phase, the voter gets a signature on a commitment to his vote from the administrator. To ensure privacy, blind signatures [14] are used, i.e. the administrator does not learn the commitment of the vote.

- Voter V selects a vote v and computes the commitment $x = \xi(v,r)$ using the commitment scheme ξ and a random key r;
- V computes the message $e = \chi(x, b)$ using a blinding function χ and a random blinding factor b;
- V digitally signs e and sends her signature $\sigma_V(e)$ to the administrator A together with her identity;
- A verifies that V has the right to vote, has not voted yet and that the signature is valid; if all these tests hold, A digitally signs e and sends his signature $\sigma_A(e)$ to V;
- V now *unblinds* $\sigma_A(e)$ and obtains $y = \sigma_A(x)$, i.e. a signed commitment to V's vote.

The second phase of the protocol is the actual voting phase.

- V sends y, A's signature on the commitment to V's vote, to the collector C using an anonymous channel;
- C checks correctness of the signature y and, if the test succeeds, enters (ℓ, x, y) into a list as an ℓ-th item.

The last phase of the voting protocol starts, once the collector decides that he received all votes, e.g. after a fixed deadline. In this phase the voters reveal the random key r which allows C to open the votes and publish them.

- C publishes the list (ℓ_i, x_i, y_i) of commitments he obtained;
- V verifies that her commitment is in the list and sends ℓ, r to C via an anonymous channel;
- C opens the ℓ-th ballot using the random r and publishes the vote v.

Note that we need to separate the voting phase into a commitment phase and an opening phase to avoid releasing partial results of the election and to ensure privacy. This is ensured by requiring synchronisation between the different agents involved in the election.

5.2 The model in applied pi

In this section we describe the applied pi calculus model of the protocol. Note that we use phase separation commands, introduced by the ProVerif tool [8] as global synchronization commands. The process first executes all instructions of a given phase before moving to the next phase. The separation of the protocol in phases is crucial for privacy to hold. As our processes do not use replication such synchronization can be easily implemented using internal communications.

Cryptographic primitives as an equational theory. We model cryptography in a Dolev-Yao style as being perfect. The equations are given below.

$$\begin{aligned}
\mathsf{open}(\mathsf{commit}(\mathsf{m},\mathsf{r}),\mathsf{r}) &= \mathsf{m}\\
\mathsf{checksign}(\mathsf{sign}(\mathsf{m},\mathsf{sk}),\mathsf{pk}(\mathsf{sk})) &= \mathsf{m}\\
\mathsf{unblind}(\mathsf{blind}(\mathsf{m},\mathsf{r}),\mathsf{r}) &= \mathsf{m}\\
\mathsf{unblind}(\mathsf{sign}(\mathsf{blind}(\mathsf{m},\mathsf{r}),\mathsf{sk}),\mathsf{r}) &= \mathsf{sign}(\mathsf{m},\mathsf{sk})
\end{aligned}$$

In this model we can note that bit commitment (modelled by the functions commit and open) is identical to classical symmetric-key encryption. For simplicity, we identify host names and public keys. Our model of cryptographic primitives is an abstraction; for example, bit commitment gives us perfect binding and hiding. Digital signatures are modeled as being signatures with message recovery, i.e. the signature itself contains the signed message which can be extracted using the checksign function. To model blind signatures we add a pair of functions blind and unblind. These functions are again similar to perfect symmetric key encryption and bit commitment. However, we add a second equation which permits us to extract a signa-

```
(* private channels *)
ν privCh.ν pkaCh1.ν pkaCh2.ν skaCh.ν skvaCh.ν skvbCh.
(* administrators *)
(processK | processA | processA | processC | processC |
(* voters *)
(let skvCh = skvaCh in let v = a in processV) |
(let skvCh = skvbCh in let v = b in processV) )
```

Process 1. Main process

ture out of a blind signature, when the blinding factor is known. Note that the equation modelling commitment cannot be applied on the term $\mathsf{open}(\mathsf{commit}(\mathsf{m}, \mathsf{r}_1), \mathsf{r}_2)$ when $\mathsf{r}_1 \neq \mathsf{r}_2$.

Process synchronisation. As mentioned, the protocol is divided into three phases, and it is important that every voter has completed the first phase before going onto the second one (and then has completed the second one before continuing to the third). We enforce this in our model by the keyword synch. When a process encounters synch n, it waits until all the other process that could encounter synch n arrive at that point too. Then all the processes are allowed to continue.

If there are k processes that can encounter synch n, we can implement the synchronisation as follows. The command synch n is replaced by $\mathsf{out}(\mathsf{n}, 0); \mathsf{in}(\mathsf{n}, =1)$ where n is a globally declared private channel. Moreover we assume an additional process $(\mathsf{in}(\mathsf{n}, =0); \ldots; \mathsf{in}(\mathsf{n}, =0); \mathsf{out}(\mathsf{n}, 1); \ldots; \mathsf{out}(\mathsf{n}, 1)$ that has k ins and k outs. This simple encoding is fine for our purpose since the value of k can be inferred by inspecting the code; it would not work if new processes were created, e.g. with "!".

Main (Process 1). The main process sets up private channels and specifies how the processes are combined in parallel. Most of the private channels are for key distribution. We only model the protocol for two voters and launch two copies of the administrator and collector process, one for each voter.

Keying material (Process 2). Our model includes a dedicated process for generating and distributing keying material modelling a PKI. Additionally, this process registers legitimate voters and also distributes the public keys of the election authorities to legitimate voters: this is modelled using restricted channels so that the attacker cannot provide false public keys.

Voter (Process 3). First, each voter obtains her secret key from the PKI as well as the public keys of the administrator. The remainder of the specification follows

```
processK=
    (* private keys *)
    ν ska. ν skva. ν skvb.
    (* corresponding public keys *)
    let (pka, pkva, pkvb)=(pk(ska), pk(skva), pk(skvb)) in
    (* public keys disclosure *)
    out(ch,pka). out(ch,pkva). out(ch,pkvb).
    (* register legitimate voters *)
    (out(privCh,pkva)|out(privCh,pkvb) |
    (* keys disclosure on private channels *)
    out(pkaCh1,pka) | out(pkaCh1,pka) | out(pkaCh2,pka) |
    out(pkaCh2,pka) | out(skaCh,ska) | out(skaCh,ska) |
    out(skvaCh,skva) | out(skvbCh,skvb))
```

Process 2. Administrator for keying material

```
processV =                    (* parameters: skvCh, v *)
    (* her private key *)
    in(skvCh,skv).
    (* public keys of the administrator *)
    in(pkaCh1,pubka).
    ν blinder. ν r.
    let committedvote = commit(v,r) in
    let blindedcommittedvote=blind(committedvote,blinder) in
    out(ch1,(pk(skv),sign(blindedcommittedvote,skv))).
    in(ch2,m2).
    let result = checksign(m2,pubka) in
    if result = blindedcommittedvote then
    let signedcommittedvote=unblind(m2,blinder) in
    synch 1.
    out(ch3,(committedvote,signedcommittedvote)).
    synch 2.
    in(ch4,(l,=committedvote,=signedcommittedvote)).
    out(ch5,(l,r))
```

Process 3. Voter process

directly the informal description given in Section 5.1.

Administrator (Process 4). The administrator first receives through a private channel his own public key as well as the public key of a legitimate voter. Legitimate voters have been registered on this private channel in Process 2 described above. The received public key has to match the voter who is trying to get a signed ballot from the administrator. If the public key indeed matches, then the administrator signs the received message which he supposes to be a blinded ballot.

```
processA =
    (* administrator's private key *)
    in(skaCh,skadm).
    (* register legimitate voters *)
    in(privCh,pubkv).
    in(ch1,m1).
    let (pubkeyv,sig) = m1 in
    if pubkeyv = pubkv then
    out(ch2,sign(checksign(sig,pubkeyv),skadm))
```

Process 4. Administrator process

```
processC =
    (* administrator's public key *)
    in(pkaCh2,pkadmin).
    synch 1. in(ch3,(m3,m4)).
    if checksign(m4,pkadmin) = m3 then
    synch 2.
    ν l.
    out(ch4,(l,m3,m4)).
    in (ch5,(=l,rand)).
    let voteV=open(m4,rand) in
    out(ch,voteV)
```

Process 5. Collector process

Collector (Process 5). When the collector receives a committed vote, he associates a fresh label 'l' with this vote. Publishing the list of votes and labels is modelled by sending those values on a public channel. Then the voter can send back the random number which served as a key in the commitment scheme together with the label. The collector receives the key matching the label and opens the vote which he then publishes.

5.3 Analysis

Vote-privacy. According to our definition, to show that the protocol respects privacy, we need to show that

$$S[V_A\{^a/_v\} \mid V_B\{^b/_v\}] \approx_\ell S[V_A\{^b/_v\} \mid V_B\{^a/_v\}]. \tag{4}$$

where $V_A = \mathsf{processV}\{^{skvaCh}/_{skvCh}\}$, $V_B = \mathsf{processV}\{^{skvbCh}/_{skvCh}\}$ and S is defined as the parallel composition of the voter processes, but with a hole instead of the two voter processes. We do not require that any of the authorities are honest, so they are not modelled in S, but rather left as part of the attacker context. To

establish this equivalence, we show that

$$\begin{array}{c} \nu\mathsf{pkaCh1}.(V_A\{^a/_v\} \mid V_B\{^b/_v\} \mid \mathsf{processK}) \\ \approx_\ell \\ \nu\mathsf{pkaCh1}.(V_A\{^b/_v\} \mid V_B\{^a/_v\} \mid \mathsf{processK}) \end{array} \tag{5}$$

Note that this implies privacy (equivalence 4) only in the case of precisely two voters (i.e., S doesn't contain any additional voters). To deduce equivalence 4 for an arbitrary context S, one would like to use the fact that labelled bisimilarity is closed under application of evaluation contexts. Unfortunately, the context $\nu\mathsf{pkaCh1}.\_$ prevents us from easily making this inference (recall that pkaCh1 is the channel on which the voters receive the public key of the administrator). Our proof is formally valid only for two voters, although a similar proof can easily be made for other numbers.

Note that to ensure privacy we do not need to require any of the keys to be secret. However, we need to ensure that both voters use the same public key for the administrator. Therefore, we send this public key on a private channel, although the secret key itself is a free name. We α-rename the bounded variables and names in the two voter processes in a straightforward way. Although ProVerif is not able to prove this observational equivalence directly, we were able to check all of the static equivalences on the frames below using ProVerif (see Lemmas 19 and 20).

We denote the left-hand process as P and the right-hand process as Q. We have that both processK start with the output of all the keys. None of these transitions depend on the value of the vote, and so they commute in the same way for P and Q. For the sake of readability, we do not detail this part. The only important point is that the output of the administrator's public key is sent on a private channel yielding an internal reduction. We have that

$$\begin{array}{l} P \xrightarrow{in(skvaCh,skva)} P_1 \xrightarrow{in(skvbCh,skvb)} P_2 \rightarrow^* \\ \quad \xrightarrow{\nu x_1.out(ch,x_1)} \nu b_A.\nu r_A.\nu b_B.\nu r_B.(P_3 \mid \{^{(pk(skva),sign(blind(commit(a,r_A),b_A),skva))}/_{x_1}\} \\ \quad \xrightarrow{\nu x_2.out(ch,x_2)} \nu b_A.\nu r_A.\nu b_B.\nu r_B.(P_4 \mid \{^{(pk(skva),sign(blind(commit(a,r_A),b_A),skva)}/_{x_1}\} \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mid \{^{(pk(skvb),sign(blind(commit(b,r_B),b_B),skvb)}/_{x_2}\}) \end{array}$$

Similarly,

$$
\begin{aligned}
Q \xrightarrow{in(skvaCh,skva)} Q_1 \xrightarrow{in(skvbCh,skvb)} Q_2 \rightarrow^* \\
\xrightarrow{\nu x_1.out(ch,x_1)} \nu b_A.\nu r_A.\nu b_B.\nu r_B.(Q_3 \mid \{^{(pk(skva),sign(blind(commit(b,r_A),b_A),skva))}/_{x_1}\} \\
\xrightarrow{\nu x_2.out(ch,x_2)} \nu b_A.\nu r_A.\nu b_B.\nu r_B.(Q_4 \mid \{^{(pk(skva),sign(blind(commit(b,r_A),b_A),skva)}/_{x_1}\} \\
\mid \{^{(pk(skvb),sign(blind(commit(a,r_B),b_B),skvb)}/_{x_2}\})
\end{aligned}
$$

We could have considered any permutation of these transitions which respects the partial order dictated by the processes. Note that for the above inputs we may consider any public term, i.e. term that does not use bounded names of the processes. For the next input of both voters, we need to consider two cases: either the input of both voters corresponds to the expected messages from the administrator or at least one input does not correspond to the correct administrator's signature. In the second case, one of the voters will block, as testing correctness of the message fails and hence they cannot synchronise. In the first case, both voters synchronise at phase 1. Until that point any move of voter $V_A\{^a/_v\}$ on the left-hand side has been imitated by voter $V_A\{^b/_v\}$ on the right-hand side and equally for the second voter. However, from now on, any move of voter $V_A\{^a/_v\}$ on the left-hand side will be matched with the corresponding move of $V_B\{^a/_v\}$ on the right-hand side and similarly for the second voter. The voters will now output the committed votes signed by the administrator. The corresponding frames are described below and are statically equivalent.

$$
\begin{aligned}
\phi_{P'} \equiv \nu b_A.\nu r_A.\nu b_B.\nu r_B.\ & \{^{(pk(skva),sign(blind(commit(a,r_A),b_A),skva))}/_{x_1}\} \mid \\
& \{^{(pk(skvb),sign(blind(commit(b,r_B),b_B),skvb))}/_{x_2}\} \mid \\
& \{^{(commit(a,r_A),sign(commit(a,r_A),ska))}/_{x_3}\} \mid \\
& \{^{(commit(b,r_B),sign(commit(b,r_B),ska))}/_{x_4}\} \\
\phi_{Q'} \equiv \nu b_A.\nu r_A.\nu b_B.\nu r_B.\ & \{^{(pk(skva),sign(blind(commit(b,r_A),b_A),skva))}/_{x_1}\} \mid \\
& \{^{(pk(skvb),sign(blind(commit(a,r_B),b_B),skvb))}/_{x_2}\} \mid \\
& \{^{(commit(a,r_B),sign(commit(a,r_B),ska))}/_{x_3}\} \mid \\
& \{^{(commit(b,r_A),sign(commit(b,r_A),ska))}/_{x_4}\}
\end{aligned}
$$

The following result can be establish using ProVerif.

Lemma 19 *The frames $\phi_{P'}$ and $\phi_{Q'}$ are statically equivalent.*

For the following input, we again consider two cases: either the input of both voters corresponds to the expected messages or at least one input does not succeed the

tests. In the second case, one of the voters will block, as testing correctness of the message fails and hence they cannot synchronise. In the first case, we obtain at the end the two frames below which are again statically equivalent.

$$\begin{aligned}\phi_{P''} \equiv \nu b_A.\nu r_A.\nu b_B.\nu r_B.\ & \left\{^{(pk(skva),sign(blind(commit(a,r_A),b_A),skva))}/_{x_1}\right\} \\ & \left\{^{(pk(skvb),sign(blind(commit(b,r_B),b_B),skvb))}/_{x_2}\right\} \mid \\ & \left\{^{(commit(a,r_A),sign(commit(a,r_A),ska))}/_{x_3}\right\} \mid \\ & \left\{^{(commit(b,r_B),sign(commit(b,r_B),ska))}/_{x_4}\right\} \mid \\ & \left\{^{(l_A,r_A)}/_{x_5}\right\} \mid \left\{^{(l_B,r_B)}/_{x_6}\right\}\end{aligned}$$

$$\begin{aligned}\phi_{Q''} \equiv \nu b_A.\nu r_A.\nu b_B.\nu r_B.\ & \left\{^{(pk(skva),sign(blind(commit(b,r_A),b_A),skva))}/_{x_1}\right\} \mid \\ & \left\{^{(pk(skvb),sign(blind(commit(a,r_B),b_B),skvb))}/_{x_2}\right\} \mid \\ & \left\{^{(commit(a,r_B),sign(commit(a,r_B),ska))}/_{x_3}\right\} \mid \\ & \left\{^{(commit(b,r_A),sign(commit(b,r_A),ska))}/_{x_4}\right\} \mid \\ & \left\{^{(l_A,r_B)}/_{x_5}\right\} \mid \left\{^{(l_B,r_A)}/_{x_6}\right\}\end{aligned}$$

Again, ProVerif is able to establish the following result.

Lemma 20 *The frames $\phi_{P''}$ and $\phi_{Q''}$ are statically equivalent.*

Note that it is sufficient to prove static equivalences for all reachable final states. Thus, Lemma 19 is actually a consequence of Lemma 20.

Note that the use of phases is crucial for privacy to be respected. When we omit the synchronisation after the registration phase with the administrator, privacy is violated. Indeed, consider the following scenario. Voter V_A contacts the administrator. As no synchronisation is considered, voter V_A can send his committed vote to the collector before voter V_B contacts the administrator. As voter V_B could not have submitted the committed vote, the attacker can link this commitment to the first voter's identity. This problem was found during a first attempt to prove the protocol where the phase instructions were omitted. The original paper divides the protocol into three phases but does not explain the crucial importance of the synchronisation after the first phase. Our analysis emphasises this need and we believe that it increases the understanding of some subtle details of the privacy property in this protocol. We may also note that we do not make any assumptions about the correctness of the administrator or the collector, who may be corrupt, However, we need to assume that both voters use the same value for the administrator's public key. Otherwise, privacy does not hold.

Receipt-freeness. This scheme is not receipt-free and was in fact not designed with receipt-freeness in mind. Indeed, if the voter gives away the random numbers for blinding and commitment, i.e. b_A and r_A, the coercer can verify that the committed vote corresponds to the coercer's wish and by unblinding the first message, the coercer can trace which vote corresponds to this particular voter. Moreover, the voter cannot lie about these values as this will immediately be detected by the coercer.

In our framework, this corresponds to the fact that there exists no V' such that:

- $V'^{\backslash out(chc,\cdot)} \approx_\ell V_A\{^a/_v\}$,
- $S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}] \approx_\ell S[V' \mid V_B\{^c/_v\}]$.

We show that there is no V' by proving that the requirements on V' are not satisfiable. We have that $V_A\{^c/_v\}^{chc}$ outputs the values r_A and b_A on the channel chc. This will generate entries in the frame. Hence, V' needs to generate similar entries in the frame. The coercer can now verify that the values r_A and b_A are used to encode the vote c in the message sent to the administrator. Thus V' is not able to commit to a value different from c, in order to satisfy the second equivalence. But then V' will not satisfy the first equivalence, since he will be unable to change his vote afterwards as the commitment to c has been signed by the administrator. Thus, the requirements on V' are not satisfiable.

The failure of receipt-freeness is not due to the possible dishonesty of the administrator or collector; even if we include them as honest parties, the protocol still doesn't guarantee receipt-freeness. It follows that coercion-resistance doesn't hold either.

6 Protocol due to Okamoto

In this section we study a protocol due to Okamoto [39] which was designed to be incoercible. However, Okamoto himself shows a flaw [40]. According to him, one of the reasons why the voting scheme he proposed had such a flaw is that no formal definition and proof of receipt-freeness and coercion-resistance have been given when the concept of receipt-freeness has been introduced by Benaloh and Tuinstra [7].

6.1 Description

The authorities managing the election are an administrator for registration, a collector for collecting the tokens and a timeliness member (denoted by T) for publishing

the final tally. The main difference with the protocol due to Fujioka *et al.* is the use of a trap-door bit commitment scheme [22] in order to retrieve receipt-freeness. Such a commitment scheme allows the agent who has performed the commitment to open it in many ways. Hence, trap-door bit commitment does not bind the voter to the vote v. Now, to be sure that the voter does not change her mind at the end (during the opening stage) she has to say how she wants to open her commitment during the voting stage. This is done by sending the required information to T through an untappable anonymous channel, i.e. a physical apparatus by which only voter V can send a message to a party, and the message is perfectly secret to all other parties.

The first phase is similar to the one of the protocol due to Fujioka *et al.*. The only change is that ξ is a trap-door bit commitment scheme.

The second phase of the protocol is the actual voting phase. Now, the voter has to say how she wants to open her commitment to the timeliness member T.

- V sends y, A's signature on the trap-door commitment to V's vote, to the collector C using an anonymous channel;
- C checks correctness of the signature y and, if the test succeeds, enters (x, y) into a list.
- V sends (v, r, x) to the timeliness member T through an untappable anonymous channel.

The last phase of the voting protocol starts, once the collector decides that he received all votes, e.g. after a fixed deadline.

- C publishes the list (x_i, y_i) of trap-door commitments he obtained;
- V verifies that her commitment is in the list;
- T publishes the list of the vote v_i in random order and also proves that he knows the permutation π and the r_i's such that $x_{\pi(i)} = \xi(v_i, r_i)$ without revealing π or the r_i's.

We have chosen to not entirely model this last phase. In particular, we do not model the zero-knowledge proof performed by the timeliness member T, as it is not relevant for illustrating our definitions of privacy, receipt-freeness and coercion-resistance. This proof of zero-knowledge is very useful to ensure that T outputs the correct vote chosen by the voter. This is important in order to ensure correctness, even in the case that T is dishonest. However, the proof of knowledge is unimportant for anonymity properties. In particular, if T is the coercer himself, then he can enforce the voter to vote as he wants as in the protocol due to Fujioka *et al.* Indeed, the timeliness member T can force the voter to give him the trap-door she has used to forge her commitment and then he can not only check if the voter has vote as he wanted, but he can also open her vote as he wants.

```
(* private channels *)
ν privCh. ν pkaCh1. ν pkaCh2.
ν skaCh. ν skvaCh. ν skvbCh. ν chT.
(* administrators *)
(processK | processA | processA | processC | processC |
 processT | processT |
(* voters *)
(let skvCh=skvaCh in let v=a in processV) |
(let skvCh=skvbCh in let v=b in processV) )
```

Process 6. Main process

6.2 The model in applied pi

Cryptographic primitives as an equational theory. The equations modelling public keys and blind signatures are the same as in Section 5.2. To model trap-door bit commitment, we consider the two following equations:

$$\mathsf{open}(\mathsf{tdcommit}(\mathsf{m},\mathsf{r},\mathsf{td}),\mathsf{r}) = \mathsf{m}$$
$$\mathsf{tdcommit}(\mathsf{m}_1,\mathsf{r},\mathsf{td}) = \mathsf{tdcommit}(\mathsf{m}_2,\mathsf{f}(\mathsf{m}_1,\mathsf{r},\mathsf{td},\mathsf{m}_2),\mathsf{td})$$

Firstly, the term $\mathsf{tdcommit}(\mathsf{m},\mathsf{r},\mathsf{td})$ models the commitment of the message m under the key r by using the trap-door td. The second equation is used to model the fact that a commitment $\mathsf{tdcommit}(\mathsf{m}_1,\mathsf{r},\mathsf{td})$ can be viewed as a commitment of any value m_2. However, to open this commitment as m_2 one has to know the key $\mathsf{f}(\mathsf{m}_1,\mathsf{r},\mathsf{td},\mathsf{m}_2)$. Note that this is possible only if one knows the key r used to forge the commitment $\mathsf{tdcommit}(\mathsf{m}_1,\mathsf{r},\mathsf{td})$ and the trap-door td.

Main (Process 6). The main process sets up private channels and specifies how the processes are combined in parallel. Most of the private channels are for key distribution. The channel chT is the untappable anonymous channel on which voters send to T how they want to open their commitment.

We have also a dedicated process for generating and distributing keying material modelling a PKI. This process is the same as the one we have given for the protocol due to Fujioka *et al.* (see Section 5).

Voter (Process 7). This process is very similar to the one given in the previous section. We use the primitive tdcommit instead of commit and at the end, the voter sends, through the channel chT, how she wants to open her commitment.

```
processV =                    (* parameters: skvCh, v *)
    (* her private key *)
    in(skvCh,skv).
    (* public keys of the administrator *)
    in(pkaCh1,pubka).
    ν blinder. ν r. ν td.
    let committedvote = tdcommit(v,r,td) in
    let blindedcommittedvote=blind(committedvote,blinder) in
    out(ch1,(pk(skv),sign(blindedcommittedvote,skv))).
    in(ch2,m2).
    let result = checksign(m2,pubka) in
    if result = blindedcommittedvote then
    let signedcommittedvote=unblind(m2,blinder) in
    synch 1.
    out(ch3,(committedvote,signedcommittedvote)).
    out(chT,(v,r,committedvote))
```

Process 7. Voter process

```
processC =
    (* administrator's public key *)
    in(pkaCh2,pkadmin).
    synch 1.
    in(ch3,(m3,m4)).
    if checksign(m4,pkadmin) = m3 then
    synch 2.
    out(chBB,(m3,m4))
```

Process 8. Collector process

Administrator. The administrator process is exactly the same as the one given in Section 5 to model the protocol due to Fujioka *et al.*

Collector (Process 8). When C receives a commitment, he checks the correctness of the signature and if he succeeds, he enters this pair into a list. This list is published in a second phase by sending the values contained in the list on the public channel chBB.

Timeliness Member (Process 9). The timeliness member receives, through chT, messages of the form $(\mathsf{vt}, \mathsf{rt}, \mathsf{xt})$ where vt is the value of the vote, xt the trap-door bit commitment and rt the key he has to use to open the commitment. In a second phase, he checks that he can obtain vt by opening the commitment xt with rt. Then, he publishes the vote vt on the board. This is modelled by sending vt on a public channel.

```
processT =
    synch 1.
    (* reception du commitment *)
    in(chT,(vt,rt,xt)).
    synch 2.
    if open(xt,rt) = vt then
    out(board, vt)
```

Process 9. Timeliness process

6.3 Analysis

Unfortunately, the equational theory which is required to model this protocol is beyond the scope of ProVerif and we cannot rely on automated verification.

Vote-privacy. Privacy can be established as in the protocol due to Fujioka *et al.* Note that the equivalence proved there does not hold here. We have to hide the outputs on the channel chT. Hence, we establish the following equivalence

$$\nu\mathsf{pkaCh1}.\nu\mathsf{chT}.(V_A\{^a/_v\} \mid V_B\{^b/_v\} \mid \mathsf{processK} \mid \mathsf{processT} \mid \mathsf{processT})$$
$$\approx_\ell$$
$$\nu\mathsf{pkaCh1}.\nu\mathsf{chT}.(V_A\{^b/_v\} \mid V_B\{^a/_v\} \mid \mathsf{processK} \mid \mathsf{processT} \mid \mathsf{processT})$$

Below we show that the protocol respects receipt-freeness and hence privacy also holds.

Receipt-freeness. To show receipt-freeness one needs to construct a process V' which successfully fakes all secrets to a coercer. The idea is for V' to vote a, but when outputting secrets to the coercer, V' lies and gives him fake secrets to pretend to cast the vote c. The crucial part is that, using trap-door commitment and thanks to the fact that the key used to open the commitment is sent through an untappable anonymous channel, the value given by the voter to the timeliness member T can be different from the one she provides to the coercer. Hence, the voter who forged the commitment, provides to the coercer the one allowing the coercer to retrieve the vote c, whereas she sends to T the one allowing her to cast the vote a.

We describe such a process V' in Process 10. To prove receipt-freeness, we need to show that

- $V'^{\backslash out(chc,\cdot)} \approx_\ell V_A\{^a/_v\}$, and
- $S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}] \approx_\ell S[V' \mid V_B\{^c/_v\}]$.

```
processV =
    (* her private key *)
    in(skvCh,skv). out(chc,skv).
    (* public keys of the administrator *)
    in(pkaCh1,pubka). out(chc,pubka).
    ν blinder. ν r. ν td.
    out(chc,blinder). out(chc,f(a,r,td,c)). out(chc,td).
    let committedvote = tdcommit(a,r,td) in
    let blindedcommittedvote=blind(committedvote,blinder) in
    out(ch1,(pk(skv),sign(blindedcommittedvote,skv))).
    out(chc,(pk(skv),sign(blindedcommittedvote,skv))).
    in(ch2,m2).
    let result = checksign(m2,pubka) in
    if result = blindedcommittedvote then
    let signedcommittedvote=unblind(m2,blinder) in
    synch 1.
    out(ch3,(committedvote,signedcommittedvote)).
    out(chc,(committedvote,signedcommittedvote)).
    out(chT,(a,r,committedvote)).
    out(chc,(c,f(a,r,td,c),committedvote))
```

Process 10. V'- Receipt-freeness

The context S we consider here is the same we have used to establish privacy, i.e. $\nu\mathsf{pkaCh1}.\nu\mathsf{chT}.(\ \_\ \mid \mathsf{processK} \mid \mathsf{processT} \mid \mathsf{processT})$; thus, as for Fujioka *et al.*, the proof is valid for two voters. The first equivalence may be seen informally by considering V' without the instructions "$\mathsf{out}(\mathsf{chc}, \ldots)$", and comparing it visually with $V_A\{^a/_v\}$. The two processes are the same.

To see the second labelled bisimulation, one can informally consider all the executions of each side. We denote the left-hand process as P and the right-hand as Q. Both processK start with the output of all the keys. For sake of readability, we ignore these outputs which are not really important for what we wish to show. We denote by $\tilde{n}$ the sequence of names $b_A, r_A, td_A, b_B, r_B, td_B$. After distribution of keying material which can be done in the same way on both sides, we observe that the instructions of $V_A\{^c/_v\}^{chc}$ can be matched with those of V'. Similarly, execution steps performed by $V_B\{^a/_v\}$ on the left are matched with $V_B\{^c/_v\}$ on the right. We need, of course, to consider all the possible executions of the two processes. However, to argue that the processes are bisimilar, we consider below a particular execution and we describe the interesting part of the two frames we obtained after execution of the first phase by the two processes.

$$
\begin{aligned}
P \xrightarrow{in(skvaCh,skva)} & \xrightarrow{\nu x_1.out(chc,x_1)} P_1 \mid \{^{skva}/_{x_1}\} \xrightarrow{in(skvbCh,skvb)} \rightarrow^* \; P_2 \mid \{^{skva}/_{x_1}\} \\
& \xrightarrow{\nu x_2.out(chc,x_2)} \xrightarrow{\nu x_3.out(chc,x_3)} \xrightarrow{\nu x_4.out(chc,x_4)} \\
& \xrightarrow{\nu x_5.out(chc,x_5)} \nu\tilde{n}.\ (P_3 \mid \{^{skva}/_{x_1}\} \mid \{^{pk(ska)}/_{x_2}\} \mid \{^{b_A}/_{x_3}\} \mid \{^{r_A}/_{x_4}\} \mid \{^{td_A}/_{x_5}\}) \\
& \xrightarrow{\nu x_6.out(ch,x_6)} \nu\tilde{n}.\ (P_4 \mid \{^{skva}/_{x_1}\} \mid \{^{pk(ska)}/_{x_2}\} \mid \{^{b_A}/_{x_3}\} \mid \{^{r_A}/_{x_4}\} \mid \{^{td_A}/_{x_5}\} \\
& \qquad \mid \{^{(pk(skva),sign(blind(tdcommit(c,r_A,td_A),b_A),skva))}/_{x_6}\}) \\
& \xrightarrow{\nu x_7.out(chc,x_7)} \nu\tilde{n}\ .(P_5 \mid \{^{skva}/_{x_1}\} \mid \{^{pk(ska)}/_{x_2}\} \mid \{^{b_A}/_{x_3}\} \mid \{^{r_A}/_{x_4}\} \mid \{^{td_A}/_{x_5}\} \\
& \qquad \mid \{^{(pk(skva),sign(blind(tdcommit(c,r_A,td_A),b_A),skva))}/_{x_6}\} \mid \{^{x_6}/_{x_7}\}) \\
& \xrightarrow{\nu x_8.out(ch,x_8)} \nu\tilde{n}\ .(P_6 \mid \{^{skva}/_{x_1}\} \mid \{^{pk(ska)}/_{x_2}\} \mid \{^{b_A}/_{x_3}\} \mid \{^{r_A}/_{x_4}\} \mid \{^{td_A}/_{x_5}\} \\
& \qquad \mid \{^{(pk(skva),sign(blind(tdcommit(c,r_A,td_A),b_A),skva))}/_{x_6}\} \mid \{^{x_6}/_{x_7}\}) \\
& \qquad \mid \{^{(pk(skvb),sign(blind(tdcommit(a,r_B,td_B),b_B),skvb))}/_{x_8}\}).
\end{aligned}
$$

Similarly,

$$
\begin{aligned}
Q \xrightarrow{in(skvaCh,skva)} & \xrightarrow{\nu x_1.out(chc,x_1)} Q_1 \mid \{^{skva}/_{x_1}\} \xrightarrow{in(skvbCh,skvb)} \rightarrow^* \; Q_2 \mid \{^{skva}/_{x_1}\} \\
& \xrightarrow{\nu x_2.out(chc,x_2)} \xrightarrow{\nu x_3.out(chc,x_3)} \xrightarrow{\nu x_4.out(chc,x_4)} \\
& \xrightarrow{\nu x_5.out(chc,x_5)} \nu\tilde{n}.\ (Q_3 \mid \{^{skva}/_{x_1}\} \mid \{^{pk(ska)}/_{x_2}\} \mid \{^{b_A}/_{x_3}\} \mid \{^{f(a,r_A,td_A,c)}/_{x_4}\} \mid \{^{td_A}/_{x_5}\}) \\
& \xrightarrow{\nu x_6.out(ch,x_6)} \nu\tilde{n}.\ (Q_4 \mid \{^{skva}/_{x_1}\} \mid \{^{pk(ska)}/_{x_2}\} \mid \{^{b_A}/_{x_3}\} \mid \{^{f(a,r_A,td_A,c)}/_{x_4}\} \mid \{^{td_A}/_{x_5}\} \\
& \qquad \mid \{^{(pk(skva),sign(blind(tdcommit(a,r_A,td_A),b_A),skva))}/_{x_6}\}) \\
& \xrightarrow{\nu x_7.out(chc,x_7)} \nu\tilde{n}\ .(Q_5 \mid \{^{skva}/_{x_1}\} \mid \{^{pk(ska)}/_{x_2}\} \mid \{^{b_A}/_{x_3}\} \mid \{^{f(a,r_A,td_A,c)}/_{x_4}\} \mid \{^{td_A}/_{x_5}\} \\
& \qquad \mid \{^{(pk(skva),sign(blind(tdcommit(a,r_A,td_A),b_A),skva))}/_{x_6}\} \mid \{^{x_6}/_{x_7}\}) \\
& \xrightarrow{\nu x_8.out(ch,x_8)} \nu\tilde{n}\ .(Q_6 \mid \{^{skva}/_{x_1}\} \mid \{^{pk(ska)}/_{x_2}\} \mid \{^{b_A}/_{x_3}\} \mid \{^{f(a,r_A,td_A,c)}/_{x_4}\} \mid \{^{td_A}/_{x_5}\} \\
& \qquad \mid \{^{(pk(skva),sign(blind(tdcommit(a,r_A,td_A),b_A),skva))}/_{x_6}\} \mid \{^{x_6}/_{x_7}\} \\
& \qquad \mid \{^{(pk(skvb),sign(blind(tdcommit(c,r_B,td_B),b_B),skvb))}/_{x_8}\}).
\end{aligned}
$$

We argue informally that the frames obtained at the end of this first phase are statically equivalent. In particular, note that the test

$$\mathsf{open}(\mathsf{unblind}(\mathsf{checksign}(\mathsf{proj}_2(\mathsf{x}_6), \mathsf{pk}(\mathsf{x}_1)), \mathsf{x}_3), \mathsf{x}_4) = \mathsf{c}$$

is true in both frames. Indeed, if we denote B' the process obtained on the left

hand-side after this first phase, we have that

$$\begin{aligned}
&\mathsf{open}(\mathsf{unblind}(\mathsf{checksign}(\mathsf{proj}_2(\mathsf{x}_6), \mathsf{pk}(\mathsf{x}_1)), \mathsf{x}_3), \mathsf{x}_4)\sigma \\
&= \mathsf{open}(\mathsf{tdcommit}(\mathsf{a}, \mathsf{r}_\mathsf{A}, \mathsf{td}_\mathsf{A}), \mathsf{f}(\mathsf{a}, \mathsf{r}_\mathsf{A}, \mathsf{td}_\mathsf{A}, \mathsf{c})) \\
&= \mathsf{open}(\mathsf{tdcommit}(\mathsf{c}, \mathsf{f}(\mathsf{a}, \mathsf{r}_\mathsf{A}, \mathsf{td}_\mathsf{A}, \mathsf{c}), \mathsf{td}_\mathsf{A}), \mathsf{f}(\mathsf{a}, \mathsf{r}_\mathsf{A}, \mathsf{td}_\mathsf{A}, \mathsf{c})) \\
&= \mathsf{c}
\end{aligned}$$

where $\phi(B') = \nu\tilde{n}.\sigma$.

For the "first input", of both voters, we need to consider two cases: either the input of both voters corresponds to the expected messages from the administrator or at least one input does not correspond to the correct administrator's signature. In the second case, one of the voters will block, as testing correctness of the message fails and hence the voters cannot synchronise. In the first case, we obtain at the end the two frames below.

$$\begin{aligned}
\phi_{P''} \equiv \nu\tilde{n}.\ & \{^{skva}/_{x_1}\} \mid \{^{pk(ska)}/_{x_2}\} \mid \{^{b_A}/_{x_3}\} \mid \{^{r_A}/_{x_4}\} \mid \{^{td_A}/_{x_5}\} \mid \\
& \{^{(pk(skva),sign(blind(tdcommit(c,r_A,td_A),b_A),skva))}/_{x_6}\} \mid \{^{x_6}/_{x_7}\} \mid \\
& \{^{(pk(skvb),sign(blind(tdcommit(a,r_B,td_B),b_B),skvb))}/_{x_8}\} \mid \\
& \{^{(tdcommit(c,r_A,td_A),sign(tdcommit(c,r_A,td_A),ska))}/_{x_9}\} \mid \{^{x_9}/_{x_{10}}\} \mid \\
& \{^{(tdcommit(a,r_B,td_B),sign(tdcommit(a,r_B,td_B),ska))}/_{x_{11}}\} \mid \\
& \{^{(c,r_A,tdcommit(c,r_A,td_A))}/_{x_{12}}\} \mid \{^{a}/_{x_{13}}\} \mid \{^{c}/_{x_{14}}\}
\end{aligned}$$

$$\begin{aligned}
\phi_{Q''} \equiv \nu\tilde{n}.\ & \{^{skva}/_{x_1}\} \mid \{^{pk(ska)}/_{x_2}\} \mid \{^{b_A}/_{x_3}\} \mid \{^{f(a,r_A,td_A,c)}/_{x_4}\} \mid \{^{td_A}/_{x_5}\} \mid \\
& \{^{(pk(skva),sign(blind(tdcommit(a,r_A,td_A),b_A),skva))}/_{x_6}\} \mid \{^{x_6}/_{x_7}\} \mid \\
& \{^{(pk(skvb),sign(blind(tdcommit(c,r_B,td_B),b_B),skvb))}/_{x_8}\} \mid \\
& \{^{(tdcommit(a,r_A,td_A),sign(tdcommit(a,r_A,td_A),ska))}/_{x_9}\} \mid \{^{x_9}/_{x_{10}}\} \mid \\
& \{^{(tdcommit(c,r_B,td_B),sign(tdcommit(c,r_B,td_B),ska))}/_{x_{11}}\} \mid \\
& \{^{(c,f(a,r_A,td_A,c),tdcommit(a,r_A,td_A))}/_{x_{12}}\} \mid \{^{a}/_{x_{13}}\} \mid \{^{c}/_{x_{14}}\}
\end{aligned}$$

We observe that the frames are statically equivalent. In particular, note that the test $\mathsf{tdcommit}(\mathsf{c}, \mathsf{x}_4, \mathsf{x}_5) = \mathsf{proj}_1(\mathsf{x}_9)$ is true in both frames and the attacker cannot distinguish the terms $\mathsf{tdcommit}(\mathsf{a}, \mathsf{r}_\mathsf{B}, \mathsf{td}_\mathsf{B})$ and $\mathsf{tdcommit}(\mathsf{c}, \mathsf{r}_\mathsf{B}, \mathsf{td}_\mathsf{B})$ since he is not able to open this commitment. As the goal of this section is to illustrate our definitions and as tool support is not provided for this equational theory we do not give a formal proof of this static equivalence.

```
processC[] =
    ν c1. ν c2. ( _ |
    (* private key of V *)
    in(c1,x1). out(chc,x1).
    (* public keys of the administrator *)
    in(c1,x2).  out(chc,x2).
    ν blinder. ν r.  ν td.
    (* nonces of V - blinder, r, td *)
    in(c1,x3). out(chc,blinder).
    in(c1,x4). out(chc,r).
    in(c1,x5). out(chc,td).

    let committedvote = tdcommit(c,r,td) in
    let blindedcommittedvote=blind(committedvote,blinder) in
    out(c2,(pk(x1),sign(blindedcommittedvote,x1))).

    (* signature of the administrator *)
    in(c1,x6). out(chc,x6).
    let result = checksign(x6,x2) in
    if result = blindedcommittedvote then
    out(c2,true).
    let signedcommittedvote=unblind(x6,blinder) in
    synch 1.
    out(c2,(committedvote,signedcommittedvote)).
    out(c2,(c,r,committedvote))
```

Process 11. Context C - coercion-resistance

Coercion-resistance. This scheme is not coercion-resistant [40]. If the coercer provides the coerced voter with the commitment that he has to use but without revealing the trap-door, the voter cannot cast her own vote a since the voter cannot produce fake outputs as she did for receipt-freeness. In terms of our definition, we need to show that there is no V' such that for all coercer C satisfying $\tilde{n} \cap fn(C) = \emptyset$ and $S[C[V_A\{^{?}/_{v}\}^{c_1,c_2}] \mid V_B\{^{a}/_{v}\} \approx_{\ell} S[V_A\{^{c}/_{v}\}^{chc} \mid V_B\{^{a}/_{v}\}]$, we have the two bullet points of the definition of coercion-resistance. Suppose V' was such a process. Let C be the context given as Process 11 (note that it is, in fact, independent of V'). In order to satisfy the second bullet point, V' has to use the commitment provided by the coercer, for otherwise this would yield an observable. But then it cannot give to the timeliness member the key to open the commitment to obtain the voter's desired vote, in order to satisfy the first bullet, since V' does not know the trap-door. Hence, for the given C, the requirements on V' are not satisfiable simultaneously.

7 Protocol due to Lee *et al.*

In this section we study a protocol based on the Lee *et al.* protocol [35]. One of the main advantages of this protocol is that it is *vote and go*: voters need to participate in the election only once, in contrast with [24] and [39] (see Sections 5 and 6), where all voters have to finish a first phase before any of them can participate in the second phase. We simplified the protocol in order to concentrate on the aspects that are important with respect to privacy, receipt-freeness and coercion-resistance. In particular we do not consider distributed authorities.

7.1 Description

The protocol relies on re-encryption and on a less usual cryptographic primitive: designated verifier proofs (DVP) of re-encryption. We start by explaining these primitives.

A re-encryption of a ciphertext (obtained using a randomised encryption scheme) changes the random coins, without changing or revealing the plaintext. In the ElGamal scheme for instance, if (x, y) is the ciphertext, this is simply done by computing (xg^r, yh^r), where r is a random number, and g and h are the subgroup generator and the public key respectively. Note that neither the creator of the original ciphertext nor the person re-encrypting knows the random coins used in the re-encrypted ciphertext, for they are a function of the coins chosen by both parties. In particular, a voter cannot reveal the coins to a potential coercer who could use this information to verify the value of the vote, by ciphering his expected vote with these coins.

A DVP of the re-encryption proves that the two ciphertexts contain indeed the same plaintext. However, a designated verifier proof only convinces one intended person, e.g., the voter, that the re-encrypted ciphertext contains the original plaintext. In particular this proof cannot be used to convince the coercer. Technically, this is achieved by giving the designated verifier the ability to simulate the transcripts of the proof. A more abstract description is the following. A DVP for a designated verifier A of a statement φ is a proof of the statement "$\varphi \vee$ I know A's private key". As A is the only one to know his own private key a proof that has not been generated by himself must be a proof of the statement φ while A himself can generate a proof of the second part of the disjunction.

Our simplified protocol can be described in three steps.

- Firstly, the voter encrypts his vote with the collector's public key (using the ElGamal scheme), signs the encrypted vote and sends it to an administrator on a private channel. The administrator checks whether the voter is a legitimate voter and has not voted yet. Then the administrator *re-encrypts* the given ciphertext,

signs it and sends it back to the voter. The administrator also provides a DVP that the two ciphertexts contain indeed the same plaintext. In practice, this first stage of the protocol can be done using a voting booth where eligibility of the voter is tested at the entrance of the booth. The booth contains a tamper-proof device which performs re-encryptions, signatures and DVP proofs.

- Then, the voter sends (via an anonymous channel) the re-encrypted vote, which has been signed by the administrator to the public board.
- Finally, the collector checks the administrator's signature on each of the votes and, if valid, decrypts the votes and publishes the final results.

7.2 The model in applied pi

Cryptographic primitives as an equational theory. The functions and equations that handle public keys and digital signature are as usual (see Section 5 for instance). To model re-encryption we add a function $\mathsf{rencrypt}$, that permits us to obtain a different encryption of the same message with another random coin which is a function of the original one and the one used during the re-encryption. We also add a pair of functions dvp and $\mathsf{checkdvp}$: dvp permits us to build a *designated verifier proof* of the fact that a message is a re-encryption of another one and $\mathsf{checkdvp}$ allows the designated verifier to check that the proof is valid. Note that $\mathsf{checkdvp}$ also succeeds for a *fake dvp* created using the designated verifier's private key. We have the following equations:

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

$$\mathsf{rencrypt}(\mathsf{penc}(\mathsf{m}, \mathsf{pk}(\mathsf{sk}), \mathsf{r1}), \mathsf{r2}) = \mathsf{penc}(\mathsf{m}, \mathsf{pk}(\mathsf{sk}), \mathsf{f}(\mathsf{r1}, \mathsf{r2}))$$

$$\mathsf{checkdvp}(\mathsf{dvp}(\mathsf{x}, \mathsf{rencrypt}(\mathsf{x}, \mathsf{r}), \mathsf{r}, \mathsf{pk}(\mathsf{sk})), \mathsf{x}, \mathsf{rencrypt}(\mathsf{x}, \mathsf{r}), \mathsf{pk}(\mathsf{sk})) = \mathsf{ok}$$

$$\mathsf{checkdvp}(\mathsf{dvp}(\mathsf{x}, \mathsf{y}, \mathsf{z}, \mathsf{skv}), \mathsf{x}, \mathsf{y}, \mathsf{pk}(\mathsf{skv})) = \mathsf{ok}$$

Main (Process 12). The main process sets up private channels and specifies how the processes are combined in parallel. Most of the private channels are for key distribution. The private channel $\mathsf{ch}\mathsf{A}_1$ (resp. $\mathsf{ch}\mathsf{A}_2$) is a private channel between the voter and her administrator. This is motivated by the fact that the administrator corresponds to a tamper-proof hardware device in this protocol. We only model the protocol for two voters and launch two copies of the administrator and collector process, one for each voter.

Keying material (Process 13). Our model includes a dedicated process for generating and distributing keying material modelling a PKI. Additionally, this process

```
    (* private channels *)
    ν privCh. ν pkaCh1. ν pkaCh2. ν pkcCh. ν skaCh. ν skcCh.
    ν skvaCh. ν skvbCh. ν chA1. ν chA2.
    (* administrators *)
    (processK | processC | processC |
    (* voters *)
    (let chA = chA1 in processA |
    (let skvCh = skvaCh in let v = a in processV)) |
    (let chA = chA2 in processA |
    (let skvCh = skvbCh in let v = b in processV)))
```

Process 12. Main process

```
processK =
    (* private key *)
    ν ska. ν skc. ν skva. ν skvb.
    (* corresponding public keys *)
    let (pka, pkc) = (pk(ska), pk(skc)) in
    let (pkva, pkvb) = (pk(skva), pk(skvb)) in
    (* publik keys disclosure *)
    out(ch,pka). out(ch,pkc). out(ch,pkva). out(ch,pkvb).
    (* register legitimate voters *)
    (out(privCh,pkva)|out(privCh,pkvb) |
    (* keys disclosure on private channels *)
    out(pkaCh,pka) | out(pkaCh,pka) | out(pkaCh,pka) |
    out(pkaCh,pka) | out(skaCh,ska) | out(skaCh,ska) |
    out(pkcCh,pkc) | out(pkcCh,pkc) | out(skcCh,skc) |
    out(skcCh,skc) | out(skvaCh,skva) | out(skvbCh,skvb))
```

Process 13. Administrator for keying material

registers legitimate voters and also distributes the public keys of the election authorities to legitimate voters: this is modelled using restricted channels so that the attacker cannot provide false public keys.

Voter (Process 14). First, each voter obtains her secret key from the PKI as well as the public keys of the election authorities. Then, a fresh random number is generated to encrypt her vote with the public key of the collector. Next, she signs the result and sends it on a private channel to the administrator. If the voter has been correctly registered, she obtains from the administrator, a re-encryption of her vote signed by the administrator together with a designated verifier proof of the fact that this re-encryption has been done correctly. If this proof is correct, then the voter sends her re-encrypted vote signed by the administrator to the collector.

Note that we used the synchronisation command to model this process. This command is crucial for privacy to hold in presence of a corrupted collector. This ensures

```
processV =                    (* parameters: skvCh, v *)
    (* her private key *)
    in(skvCh,skv).
    (* public keys of the administrators *)
    in(pkaCh1,pubka). in(pkcCh,pubkc).
    synch 1. ν r.
    let e = penc(v,pubkc,r) in
    out(chA,(pk(skv),e,sign(e,skv))).
    in(chA,m2).
    let (re,sa,dvpV) = m2 in
    if checkdvp(dvpV,e,re,pk(skv)) = ok
    then if checksign(sa,pubka) = re
    then out(ch,(re,sa))
```

Process 14. Voter process

```
processA =
    (* administrator's private key *)
    in(skaCh,skadm).
    (* register a legimitate voter *)
    in(privCh,pubkv).
    synch 1.
    in(chA,m1).
    let (pubv,enc,sig)=m1 in
    if pubv=pubkv then
    if checksign(sig,pubv)= enc
    then ν r1.
    let reAd=rencrypt(enc,r1) in
    let signAd=sign(reAd,skadm) in
    let dvpAd=dvp(enc,reAd,r1,pubv) in
    out(chA,(reAd,signAd,dvpAd))
```

Process 15. Administrator process

that key distribution is finished before any of the two voter proceeds. Otherwise an attack on privacy can be mounted since the attacker can prevent one of the voters from obtaining her keys. One may also note that this protocol is *vote and go*: even if synchronisation is used the voters participate actively only during one of the synchronised phases.

Administrator (Process 15). The administrator first receives through a private channel his own private key as well as the public key of a legitimate voter. The received public key has to match the voter who is trying to get a re-encryption of her vote signed by the administrator. The administrator has also to prove to the voter that he has done the re-encryption properly. For this, he builds a designated verifier proof which will be only convincing for the voter.

```
processC =
    (* collector's private key *)
    in(skcCh,privc).
    (* administrator's public key *)
    in(pkaCh2,pkadmin).
    synch 1.
    in(ch,m3).
    let (ev,sev) = m3 in
    if checksign(sev,pkadmin) = ev
    then let voteV = decrypt(ev,privc) in
    synch 2.
    out(ch,voteV)
```

Process 16. Collector process

Collector (Process 16). First, the collector receives all the signed ballots. He checks the signature and decrypts the result with his private key to obtain the value of the vote in order to publish the results. Although it is not mentioned in the description of the protocol [35], it seems reasonable to think that the collector does not accept the same ballot twice. For sake of readability, we do not model this feature in Process 16; however, we will model it when we come to receipt-freeness, since it is crucial there. Finally, when all votes have been submitted to the collector (synchronisation is achieved using the synchronisation instruction), they are published.

7.3 Analysis

Let $V_A = V\{^{skvaCh}/_{skvCh}\}\{^{chA1}/_{chA}\}$ and $V_B = V\{^{skvbCh}/_{skvCh}\}\{^{chA2}/_{chA}\}$. Note that again we have to establish all the equivalences manually: ProVerif is not able to deal with equational theories such as this one.

Vote privacy. We show that the protocol respects privacy. For this, we establish the following equivalence

$$S[V_A\{^a/_v\} \mid V_B\{^b/_v\}] \approx_\ell S[V_A\{^b/_v\} \mid V_B\{^a/_v\}]$$

where $S = \nu \mathsf{pkaCh1}, \mathsf{pkcCh}, \mathsf{skaCh}, \mathsf{chA1}, \mathsf{chA2}.(\ \_\ \mid \mathsf{processK} \mid \mathsf{processA}\{^{\mathsf{chA1}}/_{\mathsf{chA}}\} \mid \mathsf{processA}\{^{\mathsf{chA2}}/_{\mathsf{chA}}\})$

As for the other case studies, we prove orivacy only for the case of two voters.

Privacy does not require any of the keys to be secret. However, we need to ensure that both voters use the same public key for the administrator and for the collector. Therefore, we send public keys on a private channel, although the corresponding private keys can be considered as free names. We assume that both administrators have the same private key and that both voters have the right to vote. If any of these conditions is not satisfied, privacy does not hold.

We denote the left-hand process as P and the right-hand process as Q. The process$\mathsf{K}$ starts with the output of all the keys. For the sake of readability, we ignore some of these outputs which are not important for our analysis and we write $\nu\tilde{r}$ instead of the sequence $\nu r_A.\nu r_B.\nu r_1.\nu r_2$.

$$
\begin{aligned}
P \xrightarrow{in(skvaCh,skva)} \to^* \xrightarrow{in(skvbCh,skvb)} \to^* P_1 \\
\xrightarrow{\nu x_1.out(ch,x_1)} \nu\tilde{r}.(P_2 \mid \{^{(penc(a,pkc,f(r_A,r_1)),sign(penc(a,pkc,f(r_A,r_1)),ska)}/_{x_1}\} \\
\xrightarrow{\nu x_2.out(ch,x_2)} \nu\tilde{r}.(P_3 \mid \{^{(penc(a,pkc,f(r_A,r_1)),sign(penc(a,pkc,f(r_A,r_1)),ska)}/_{x_1}\} \\
\mid \{^{(penc(b,pkc,f(r_B,r_2)),sign(penc(b,pkc,f(r_B,r_2)),ska)}/_{x_2}\})
\end{aligned}
$$

Similarly,

$$
\begin{aligned}
Q \xrightarrow{in(skvaCh,skva)} \to^* \xrightarrow{in(skvbCh,skvb)} \to^* Q_1 \\
\xrightarrow{\nu x_1.out(ch,x_1)} \nu\tilde{r}.(Q_2 \mid \{^{(penc(a,pkc,f(r_B,r_2)),sign(penc(a,pkc,f(r_B,r_2)),ska)}/_{x_1}\} \\
\xrightarrow{\nu x_2.out(ch,x_2)} \nu\tilde{r}.(Q_3 \mid \{^{(penc(a,pkc,f(r_B,r_2)),sign(penc(a,pkc,f(r_B,r_2)),ska)}/_{x_1}\} \\
\mid \{^{(penc(b,pkc,f(r_A,r_1)),sign(penc(b,pkc,f(r_A,r_1)),ska)}/_{x_2}\})
\end{aligned}
$$

The resulting frames are statically equivalent. Note that, during key distribution, the process $V_A\{^a/_v\}$ is matched with $V_A\{^b/_v\}$, while afterwards $V_A\{^a/_v\}$ is matched with $V_B\{^a/_v\}$. Therefore, we require a phase after the keying distribution.

Receipt-freeness. To show receipt-freeness one needs to construct a process V' which can successfully fake all secrets to a coercer. The idea is that V' votes a, but when outputting secrets to the coercer V' prepares all outputs as if she was voting c. The crucial part is that, using her private key, she provides a fake DVP stating that the actual re-encryption of the encryption of vote a is a re-encryption of the encryption of vote c. Given our equational theory, the two resulting frames are statically equivalent because for both the real and the fake DVP, checkdvp returns ok.

To establish receipt-freeness, we have to assume that the collector is trusted. Indeed, it is important to be sure that its private key remains secret. Otherwise, an attack against receipt-freeness can be mounted: if the coercer knows the collector's

```
processV'=
    (* her private key *)
    in(skvaCh,skv). out(chc,skv).
    (* public keys of administrators *)
    in(pkaCh,pubka). out(chc,pubka).
    in(pkcCh,pubkc). out(chc,pubkc).
    synch 1.
    ν r. out(chc,r).
    let e = penc(a,pubkc,r) in
    out(chA1,(pk(skv),e,sign(e,skv))).

    (* message from the administrator *)
    in(chA1,m2).
    let (re,sa,dvpV) = m2 in
    if checkdvp(dvpV,e,re,pk(skv))=ok then
    ν r'.
    let fk=dvp(penc(c,pubkc,r), re,r',skv) in
    out(chc,(re,sa,fk)).
    if checksign(sa,pubka) = re then
    out(ch,(re,sa))
```

Process 17. Process V' - Receipt-Freeness

private key he can directly decrypt the re-encryption and check whether the vote is c rather than relying on the designated verifier proof. Note that, in reality [35], a threshold encryption scheme is used and decryption has to be performed by multiple collectors. Hence, their scheme can deal with some corrupt collectors. It is also important that the private key of the administrator remains secret. Otherwise an attacker can forge any vote and submit it to the collector.

Process 17 shows a possible V'. To prove receipt-freeness, we need to show

- $V'^{\backslash out(chc,\cdot)} \approx_\ell V_A\{^a/_v\}$, and
- $S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}] \approx_\ell S[V' \mid V_B\{^c/_v\}]$.

where S represents all of the remaining process.

The first labelled bisimulation may be seen informally by considering V' with the "out$(chc, ...)$" commands removed, and comparing it visually with V_A. To see the second labelled bisimulation, one can informally consider all the executions of each side. S consists of the Main process, and therefore includes processK, the two processA's, and the two processC's, but it has a hole for the two voter processes. As shown above, the hole is filled by $V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}$ on the left and by $V' \mid V_B\{^c/_v\}$ on the right. Executions of $V_A\{^c/_v\}^{chc}$ are matched with those of V'; similarly, $V_B\{^a/_v\}$ on the left is matched with $V_B\{^c/_v\}$ on the right. To illustrate this, we consider a particular execution on the left, and we give the corresponding execution on the right. Here the process P_1 is the one obtained after key

distribution. The sequence of names $\tilde{n}$ denotes r_A, r_1, r_B, r_2, r' and also $skvb, skc$ and ska but not $skva$ (coerced voter). We write $pkva$ instead of $pk(skva)$ and assume that public keys are in the frame. We denote by $p_A = penc(c, pkc, f(r_A, r_1))$ and by $p_B = penc(a, pkc, f(r_B, r_2))$.

$$
\begin{aligned}
P_1 &\xrightarrow{\nu x_1.out(ch,x_1)} \nu\tilde{n}.(P_2 \mid \{^{r_A}/_{x_1}\}) \\
&\xrightarrow{\nu x_2.out(ch,x_2)} \nu\tilde{n}.(P_3 \mid \{^{r_A}/_{x_1}\} \mid \{^{(p_A,sign(p_A,ska),dvp(penc(c,pkc,r_A),p_A,r_1,pkva))}/_{x_2}\}) \\
&\xrightarrow{\nu x_3.out(ch,x_3)} \nu\tilde{r}.(P_4 \mid \{^{r_A}/_{x_1}\} \mid \{^{(p_A,sign(p_A,ska),dvp(penc(c,pkc,r_A),p_A,r_1,pkva))}/_{x_2}\} \\
&\qquad\qquad \mid \{^{(p_A,sign(p_A,ska)}/_{x_3}\}) \\
&\xrightarrow{\nu x_4.out(ch,x_4)} \nu\tilde{n}.(P_5 \mid \{^{r_A}/_{x_1}\} \mid \{^{(p_A,sign(p_A,ska),dvp(penc(c,pkc,r_A),p_A,r_1,pkva))}/_{x_2}\} \\
&\qquad\qquad \mid \{^{(p_A,sign(p_A,ska)}/_{x_3}\} \mid \{^{(p_B,sign(p_B,ska))}/_{x_4}\})
\end{aligned}
$$

Similarly, we have that

$$
\begin{aligned}
Q_1 &\xrightarrow{\nu x_1.out(ch,x_1)} \nu\tilde{n}.(Q_2 \mid \{^{r_A}/_{x_1}\}) \\
&\xrightarrow{\nu x_2.out(ch,x_2)} \nu\tilde{n}.(Q_3 \mid \{^{r_A}/_{x_1}\} \mid \{^{(q_A,sign(q_A,ska),dvp(penc(c,pkc,r_A),q_A,r',skva))}/_{x_2}\}) \\
&\xrightarrow{\nu x_3.out(ch,x_3)} \nu\tilde{n}.(Q_4 \mid \{^{r_A}/_{x_1}\} \mid \{^{(q_A,sign(q_A,ska),dvp(penc(c,pkc,r_A),q_A,r',skva))}/_{x_2}\} \\
&\qquad\qquad \mid \{^{(q_A,sign(q_A,ska)}/_{x_3}\}) \\
&\xrightarrow{\nu x_4.out(ch,x_4)} \nu\tilde{n}.(Q_5 \mid \{^{r_A}/_{x_1}\} \mid \{^{(q_A,sign(q_A,ska),dvp(penc(c,pkc,r_A),q_A,r',skva))}/_{x_2}\} \\
&\qquad\qquad \mid \{^{(q_A,sign(q_A,ska)}/_{x_3}\} \mid \{^{(q_B,sign(q_B,ska))}/_{x_4}\})
\end{aligned}
$$

where $q_A = penc(a, pkc, f(r_A, r_1))$ and $q_B = penc(c, pkc, f(r_B, r_2))$.

Note that, the test $\mathsf{checkdvp}(\mathsf{proj_3}(\mathsf{x_2}), \mathsf{penc}(\mathsf{c}, \mathsf{pkc}, \mathsf{x_1}), \mathsf{proj_1}(\mathsf{x_2}), \mathsf{pk}(\mathsf{skva})) = \mathsf{ok}$ is true in both frames. Now, for the input of the collector, we have to consider any public terms. There are essentially two cases. Either the input of both collectors corresponds to the votes submitted by both voters or at least one of the inputs does not. In the last case, since the attacker is not able to provide fake inputs of the expected form, i.e. the input needs to be signed by the administrator, this means that either the collector will block or that both inputs are exactly the same. To prevent the last case, we have to ensure that the collector does not accept a same vote twice. This can be modelled by adding a process in charge of checking double votes and by slightly modifying the processC. The additional process is described in Process 18. In the collector process we add the following instructions just before "synch 2": $\mathsf{out}(\mathsf{privDblChk}, \mathsf{ballot}).\mathsf{in}(\mathsf{privDblChk}, \mathsf{x})$. if $\mathsf{x} = \mathsf{ok}$ then $[\ldots]$ where privDblChk is a restricted channel.

```
doubleCheck =
    in(privDblChk, ballot1). out(privDblChk, ok).
    in(privDblChk, ballot2).
    if ballot1=ballot2 then 0 else out(privDblChk, ok)
```

Process 18. Process to prevent double ballot

We know that if the tests succeeded, both collectors synchronise at phase 2. Up to that point any move of the collector that received the vote of $V_A\{^c/_v\}^{chc}$ on the left-hand side has been imitated on the right-hand side by the collector that received the vote of the voter $V_B\{^c/_v\}$, and similarly for the second collector. The interesting part of the frames obtained after a complete execution is described below.

$$\phi_{P'} \equiv \nu\tilde{n}.\ (\{^{r_A}/_{x_1}\} \mid \{^{(p_A,sign(p_A,ska),dvp(penc(c,pkc,r_A),p_A,r_1,pkva))}/_{x_2}\} \mid \{^{(p_A,sign(p_A,ska)}/_{x_3}\} \mid \{^{(p_B,sign(p_B,ska))}/_{x_4}\} \mid \{^{a}/_{x_5}\} \mid \{^{c}/_{x_6}\})$$

$$\phi_{Q'} \equiv \nu\tilde{n}.\ (\{^{r_A}/_{x_1}\} \mid \{^{(q_A,sign(q_A,ska),dvp(penc(c,pkc,r_A),q_A,r',skva))}/_{x_2}\} \mid \{^{(q_A,sign(q_A,ska)}/_{x_3}\} \mid \{^{(q_B,sign(q_B,ska))}/_{x_4}\} \mid \{^{a}/_{x_5}\} \mid \{^{c}/_{x_6}\})$$

Coercion-resistance. We prove coercion resistance by constructing V', which is similar to the one for receipt-freeness. However, for coercion-resistance the coercer also provides the inputs for the messages to send out. Thanks to the fact that

$$S[C[V_A\{^?/_v\}^{c_1,c_2}] \mid V_B\{^a/_v\}] \approx_\ell S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}],$$

we know that the coercer prepares messages corresponding to the given vote c. Hence,

- V' fakes the outputs as in the case of receipt-freeness; the non-coerced voter will counter-balance the outcome, by choosing the vote c;
- V' simply ignores the inputs provided by the coercer.

Such a process V' is shown in Process 19. Similar reasoning to the one used above (for receipt freeness) can be used here, to establish that the conditions

- $C[V']^{\setminus out(chc,\cdot)} \approx_\ell V_A\{^a/_v\}$
- $S[C[V_A\{^?/_v\}^{c_1,c_2} \mid V_B\{^a/_v\}] \approx_\ell S[C[V'] \mid V_B\{^c/_v\}]$,

hold, thus establishing coercion resistance. It is a bit more difficult to perform this reasoning since we have to consider any context $C = \nu c_1.\nu c_2.(\_ \mid P)$ such that $\tilde{n} \cap fn(C) = \emptyset$ and $S[C[V_A\{^?/_v\}^{c_1,c_2}] \mid V_B\{^a/_v\}] \approx_\ell S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}]$.

For the first condition, we can see that if the process $C[V']^{\setminus out(chc,\cdot)}$ does not block then it has the same behaviour as $V_A\{^a/_v\}$ since V' completely ignores the inputs provided by C. The only point is to ensure that V' can fake the outputs to C as in the case of receipt-freeness. This is indeed possible to do so since the voter does not have to know any private data used by the coercer to prepare the messages. (For instance, the voter does not have to know the nonce used by the coercer when he encrypts the vote c.)

To obtain the second condition, it is sufficient to show that the equivalence

$$S[V'' \mid V_B\{^c/_v\}] \approx_\ell S[C[V'] \mid V_B\{^c/_v\}]$$

holds, where V'' is the process provided for receipt-freeness (Process 17). Note that the processes $C[V']$ and V'' are not bisimilar by themselves, because some tests involving messages outputted on chA1 allows us to distinguish them. Indeed, it may be possible that the coercer (i.e. the context C) chooses to generate his own nonce r_c to encrypt his vote c and does not use the one provided by the voter. In such a case, the coercer has to output r_c on the channel chc, and does not forward the nonce provided by the voter, in order to ensure that $S[C[V_A\{^?/_v\}^{c_1,c_2}] \mid V_B\{^a/_v\}] \approx_\ell S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}]$. This means that the outputs performed on chc by V'' on the left hand-side and by the coercer C on the right hand-side are not quite the same. However, those tests cannot be performed when these processes are put inside the context S, because chA1 is restricted.

8 Conclusion

We have defined a framework for modelling cryptographic voting protocols in the applied pi calculus, and shown how to express in it the properties of vote-privacy, receipt-freeness and coercion-resistance. Within the framework, we can stipulate which parties are assumed to be trustworthy in order to obtain the desired property. We investigated three protocols from the literature. Our results are summarised in Figure 1.

We have proved the intuitive relationships between the three properties: for a fixed set of trusted authorities, coercion-resistance implies receipt-freeness, and receipt-freeness implies vote-privacy.

Our definition of coercion-resistance does not attempt to handle "fault attacks", in which the coercer supplies material which forces the voter to vote randomly, or to vote incorrectly resulting in an abstention (these attacks are respectively called *randomisation* and *forced abstention* attacks in the work of Juels *et al.* [31]). A protocol which succumbs to such attacks could still be considered coercion-resistant according to our definition. In our model, the coercer can count the votes for each candidate, so it seems to be in fact impossible to resist fault attacks fully.

```
processV'=
    (* her private key *)
    in(skvaCh,skv). out(c1,skv).
    (* public keys of administrators *)
    in(pkaCh,pubka). out(c1,pubka).
    in(pkcCh,pubkc). out(c1,pubkc).
    synch 1.
    ν r. out(c1,r).
    let e = penc(a,pubkc,r) in
    (* instruction from the coercer *)
    in(c2,x1).
    let (pi,ei,si) = x1 in
    out(chA1,(pk(skv),e,sign(e,skv))).

    (* message from the administrator *)
    in(chA1,m2).
    let (re,sa,dvpV) = m2 in
    if checkdvp(dvpV,e,re,pk(skv)) = ok then
    ν r'.
    let fk = dvp(ei,re,r',skv) in
    out(c1,(re,sa,fk)).
    if checksign(sa,pubka) = re then
    in(c2,x2). out(ch,(re,sa))
```

Process 19. Process V' - coercion-resistance

Property	*Fujioka et al.*	*Okamoto et al.*	*Lee et al.*
Vote-privacy	✓	✓	✓
trusted authorities	none	timeliness mbr.	administrator
Receipt-freeness	×	✓	✓
trusted authorities	n/a	timeliness mbr.	admin. & collector
Coercion-resistance	×	×	✓
trusted authorities	n/a	n/a	admin. & collector

Fig. 1: Summary of protocols and properties

Our reasoning about bisimulation in applied pi is rather informal. In the future, we hope to develop better techniques for formalising and automating this reasoning. The ProVerif tool goes some way in this direction, but the technique it uses is focused on process which have the same structure and differ only in the choice of terms [9]. The sort of reasoning we need in this paper often involves a bisimulation relation which does not follow the structure of the processes. For example, in proving vote-privacy for Fujioka *et al.*, early on we match $V_A\{^a/_v\}$ on the left-hand side with $V_A\{^b/_v\}$ on the right-hand side, while later we match $V_A\{^a/_v\}$ on the left

with $V_B\{^a/_v\}$ on the right. It would be useful to automate this kind of reasoning, or to investigate more general and more powerful methods for establishing bisimulation. Symbolic reasoning has proved successful for reachability properties [37,5], in which terms input from the environment are represented as symbolic variables, together with some constraints. One direction we are investigating is the development of symbolic bisimulation and corresponding decision procedures for the finite applied pi calculus. This work has been initiated in [19].

Our definition of coercion-resistance involves quantification over all possible contexts which satisfy a certain condition, and this makes it hard to work with in practice. Coercion-resistance may thus be seen as a kind of observational equivalence but with a restriction on the powers of the observer. Our earlier paper [18] included a notion which we called *adaptive simulation*, a variant of bisimulation which attempts to model the coerced voter's ability to adapt her vote according to the instructions of the coercer. Unfortunately, we have found this notion to have some undesirable properties, and we have not used it in this paper. In the future, we hope to find a corresponding restriction of labelled bisimilarity, which will help us to reason with coercion-resistance more effectively.

Acknowledgments Michael Clarkson read our CSFW paper [18] and asked us several challenging questions, which were instrumental in helping us prepare this paper. Anonymous reviewers of this journal article provided many detailed comments which were very useful in helping us to improve its quality.

References

[1] Martín Abadi, Bruno Blanchet, and Cédric Fournet. Just fast keying in the pi calculus. In *Proc. 13th European Symposium on Programming (ESOP'04)*, volume 2986 of *LNCS*, pages 340–354. Springer, 2004.

[2] Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *Proc. 28th ACM Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115, London, UK, 2001. ACM.

[3] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Proc. 4th ACM Conference on Computer and Communications Security (CCS'97)*, pages 36–47. ACM Press, 1997.

[4] A. Baskar, R. Ramanujam, and S.P. Suresh. Knowledge-based modelling of voting protocols. In *Proc. 11th Conference on Theoretical Aspects of Rationality and Knowledge (TARK'07)*, pages 62–71, 2007.

[5] Mathieu Baudet. Deciding security of protocols against off-line guessing attacks. In *Proc. 12th ACM Conference on Computer and Communications Security (CCS'05)*, pages 16–25, Alexandria, Virginia, USA, 2005. ACM Press.

[6] Josh Benaloh. *Verifiable Secret Ballot Elections*. PhD thesis, Yale University, 1987.

[7] Josh Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *Proc. 26th Symposium on Theory of Computing (STOC'94)*, pages 544–553. ACM Press, 1994.

[8] Bruno Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *Proc. 14th IEEE Computer Security Foundations Workshop (CSFW'01)*, pages 82–96. IEEE Comp. Soc. Press, 2001.

[9] Bruno Blanchet, Martín Abadi, and Cédric Fournet. Automated Verification of Selected Equivalences for Security Protocols. In *Proc. 20th IEEE Symposium on Logic in Computer Science (LICS 2005)*, pages 331–340. IEEE Comp. Soc. Press, 2005.

[10] Ran Canetti and Rosario Gennaro. Incoercible multiparty computation. In *Proc. 37th Symposium on Foundations of Computer Science (FOCS'96)*, pages 504–513. IEEE Comp. Soc. Press, 1996.

[11] Konstantinos Chatzikokolakis and Catuscia Palamidessi. Probable innocence revisited. In *Proc. 3rd Formal Aspects in Security and Trust (FAST'05)*, volume 3866 of *LNCS*, pages 142–157. Springer, 2006.

[12] Konstantinos Chatzikokolakis, Catuscia Palamidessi, and P. Panangaden. Anonymity protocols as noisy channels. In *Proc. 2nd Symposium on Trustworthy Global Computing (TGC'06)*, LNCS. Springer, 2006. To appear.

[13] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, February 1981.

[14] David Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology – CRYPTO'82*, pages 199–203. Plenum Press, 1983.

[15] David Chaum. Elections with unconditionally-secret ballots and disruption equivalent to breaking RSA. In *Advances in Cryptology – Eurocrypt'88*, volume 330 of *LNCS*, pages 177–182. Springer, 1988.

[16] David Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security and Privacy*, 2(1):38–47, 2004.

[17] David Chaum, Peter Y. A. Ryan, and Steve Schneider. A practical, voter-verifiable election scheme. In *Proc. 10th European Symposium On Research In Computer Security (ESORICS'05)*, volume 3679 of *LNCS*, pages 118–139. Springer, 2005.

[18] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Coercion-resistance and receipt-freeness in electronic voting. In *Proc. 19th Computer Security Foundations Workshop (CSFW'06)*, pages 28–39. IEEE Comp. Soc. Press, 2006.

[19] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Symbolic bisimulation for the applied pi-calculus. In *Proc. 27th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'07)*, LNCS. Springer, 2007. To appear.

[20] Claudia Díaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards measuring anonymity. In *Proc. 2nd International Workshop on Privacy Enhancing Technologies (PET'02)*, volume 2482 of *LNCS*, pages 54–68. Springer, 2002.

[21] Ariel J. Feldman, J. Alex Halderman, and Edward W. Felten. Security analysis of the diebold accuvote-ts voting machine. `http://itpolicy.princeton.edu/voting/`, 2006.

[22] Marc Fischlin. *Trapdoor Commitment Schemes and Their Applications*. PhD thesis, Fachbereich Mathematik Johann Wolfgang Goethe-Universität Frankfurt am Main, 2001.

[23] Cédric Fournet and Martín Abadi. Hiding names: Private authentication in the applied pi calculus. In *Proc. International Symposium on Software Security (ISSS'02)*, volume 2609 of *LNCS*, pages 317–338. Springer, 2003.

[24] Atsushi Fujioka, Tatsuaki Okamoto, and Kazui Ohta. A practical secret voting scheme for large scale elections. In *Advances in Cryptology – AUSCRYPT '92*, volume 718 of *LNCS*, pages 244–251. Springer, 1992.

[25] Rop Gonggrijp, Willem-Jan Hengeveld, Andreas Bogk, Dirk Engling, Hannes Mehnert, Frank Rieger, Pascal Scheffers, and Barry Wels. Nedap/Groenendaal ES3B voting computer: a security analysis. `www.wijvertrouwenstemcomputersniet.nl/other/es3b-en.pdf`. Retrieved 24 October 2007.

[26] Joseph Y. Halpern and Kevin R. O'Neill. Anonymity and information hiding in multiagent systems. *Journal of Computer Security*, 13(3):483–512, 2005.

[27] Martin Hirt and Kazue Sako. Efficient receipt-free voting based on homomorphic encryption. In *Advances in Cryptography – Eurocrypt'00*, volume 1807 of *LNCS*, pages 539–556. Springer, 2000.

[28] Hugo L. Jonker and Erik P. de Vink. Formalising Receipt-Freeness. In *Proc. Information Security (ISC'06)*, volume 4176 of *LNCS*, pages 476–488. Springer, 2006.

[29] Hugo L. Jonker and Wolter Pieters. Receipt-freeness as a special case of anonymity in epistemic logic. In *Proc. AVoSS Workshop On Trustworthy Elections (WOTE'06)*, 2006.

[30] Wen-Shenq Juang and Chin-Laung Lei. A secure and practical electronic voting scheme for real world environments. *IEICE Transaction on Fundamentals of Electronics, Communications and Computer Science, E80A*, 1:64–71, January 1997.

[31] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In *Proc. Workshop on Privacy in the Electronic Society (WPES'05)*. ACM Press, 2005.

[32] Detlef Kähler, Ralf Küsters, and Thomas Wilke. A Dolev-Yao-based Definition of Abuse-free Protocols. In *Proc. 33rd International Colloqium on Automata, Languages, and Programming (ICALP'06)*, volume 4052 of *LNCS*, pages 95–106. Springer, 2006.

[33] Tadayoshi Kohno, Adam Stubblefield, Aviel D. Rubin, and Dan S. Wallach. Analysis of an electronic voting system. In *Proc. 25th IEEE Symposium on Security and Privacy (SSP'04)*, pages 27–28. IEEE Comp. Soc. Press, 2004.

[34] Steve Kremer and Mark D. Ryan. Analysis of an electronic voting protocol in the applied pi-calculus. In *Proc. 14th European Symposium On Programming (ESOP'05)*, volume 3444 of *LNCS*, pages 186–200. Springer, 2005.

[35] Byoungcheon Lee, Colin Boyd, Ed Dawson, Kwangjo Kim, Jeongmo Yang, and Seungjae Yoo. Providing receipt-freeness in mixnet-based voting protocols. In *Proc. Information Security and Cryptology (ICISC'03)*, volume 2971 of *LNCS*, pages 245–258. Springer, 2004.

[36] Sjouke Mauw, Jan H.S. Verschuren, and Erik P. de Vink. A formalization of anonymity and onion routing. In *Proc. 9th European Symposium on Research Computer Security (ESORICS'04)*, volume 3193 of *LNCS*, pages 109–124. Springer, 2004.

[37] Jonathan K. Millen and Vitaly Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proc. 8th ACM Conference on Computer and Communications Security (CCS'01)*, pages 166–175. ACM Press, 2001.

[38] Christoffer Rosenkilde Nielsen, Esben Heltoft Andersen, and Hanne Riis Nielson. Static analysis of a voting protocol. In *Proc. 2nd Workshop on Automated Reasoning for Security Protocol Analysis (ARSPA'05)*, 2005.

[39] Tatsuaki Okamoto. An electronic voting scheme. In *Proc. IFIP World Conference on IT Tools*, pages 21–30, 1996.

[40] Tatsuaki Okamoto. Receipt-free electronic voting schemes for large scale elections. In *Proc. 5th Int. Security Protocols Workshop*, volume 1361 of *LNCS*, pages 25–35. Springer, 1997.

[41] Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for web transactions. *ACM Trans. Inf. Syst. Secur.*, 1(1):66–92, 1998.

[42] Steve Schneider and Abraham Sidiropoulos. CSP and anonymity. In *Proc. 4th European Symposium On Research In Computer Security (ESORICS'96)*, volume 1146 of *LNCS*, pages 198–218. Springer, 1996.

[43] Andrei Serjantov and George Danezis. Towards an information theoretic metric for anonymity. In *Proc. 2nd International Workshop on Privacy Enhancing Technologies (PET'02)*, volume 2482 of *LNCS*, pages 41–53. Springer, 2002.

[44] Vitaly Shmatikov. Probabilistic analysis of anonymity. In *Proc. 15th Computer Security Foundations Workshop (CSFW'02)*, pages 119–128. IEEE Comp. Soc. Press, 2002.

[45] Paul F. Syverson, David M. Goldschlag, and Michael G. Reed. Anonymous connections and onion routing. In *Proc. 18th IEEE Symposium on Security and Privacy (SSP'97)*, pages 44–54. IEEE Comp. Soc. Press, 1997.

Appendix A Proof of Lemma 14

Lemma 14 *Let P be a closed plain process and ch a channel name such that $ch \notin fn(P) \cup bn(P)$. We have $(P^{ch})^{\backslash out(ch,\cdot)} \approx_\ell P$.*

PROOF. Let P be a closed plain process. We show by induction on the size of P that for any channel name ch such that $ch \notin fn(P) \cup bn(P)$ we have $P^{ch\backslash out(ch,\cdot)} \approx_\ell P$. The size of the null process is defined to be 0. Prefixing the process P by a restriction, an input or an output or putting it under a replication adds 1 to its size. The size of the process $P \mid Q$ (resp. if $M = N$ then P else Q) is the sum of the size of P and Q plus 1.

The base case where $P = 0$ is trivial. Let ch be a channel name such that $ch \notin fn(P) \cup bn(P)$. The possibilities for building P are the following:

- $P = P_1 \mid P_2$. In such a case, we have:

$$\begin{aligned}
P^{ch\backslash out(ch,.)} &\mathrel{\widehat{=}} (P_1{}^{ch} \mid P_2{}^{ch})^{\backslash out(ch,.)} \\
&\mathrel{\widehat{=}} \nu ch.(P_1{}^{ch} \mid P_2{}^{ch} \mid !\mathsf{in}(ch,x)) \\
&\approx_\ell \nu ch.(P_1{}^{ch} \mid !\mathsf{in}(ch,x)) \mid \nu ch.((P_2)^{ch} \mid !\mathsf{in}(ch,x)) \\
&\qquad\qquad \text{since } \mathsf{in}(ch,.) \text{ occurs neither in } P_1^{ch} \text{ nor in } P_2^{ch} \\
&\approx_\ell P_1^{ch\backslash out(ch,.)} \mid P_2^{ch\backslash out(ch,.)} \\
&\approx_\ell P_1 \mid P_2 \qquad\qquad \text{by induction hypothesis} \\
&= P
\end{aligned}$$

- $P = \nu n.P_1$. We have:

$$\begin{aligned}
P^{ch\backslash out(ch,.)} &= (\nu n.P_1)^{ch\backslash out(ch,.)} \\
&\mathrel{\widehat{=}} \nu ch.(\nu n.\mathsf{out}(ch,n).P_1{}^{ch} \mid !\mathsf{in}(ch,x)) \\
&\approx_\ell \nu ch.(\nu n.P_1{}^{ch} \mid !\mathsf{in}(ch,x)) \\
&\equiv \nu n.\nu ch.(P_1{}^{ch} \mid !\mathsf{in}(ch,x)) \qquad\qquad \text{since } n \neq ch \\
&\mathrel{\widehat{=}} \nu n.P_1{}^{ch\backslash out(ch,.)} \\
&\approx_\ell \nu n.P_1 \qquad\qquad \text{by induction hypothesis} \\
&= P
\end{aligned}$$

- $P = \mathsf{in}(c, y).P_1$. Note that $c \neq ch$. We have:

$$\begin{aligned}
P^{ch\backslash out(ch,.)} &= (\mathsf{in}(c, y).P_1)^{ch\backslash out(ch,.)} \\
&\widehat{=} \ \nu ch.(\mathsf{in}(c, y).\mathsf{out}(ch, y).P_1^{ch} \mid !\mathsf{in}(ch, x)) \\
&\approx_\ell \mathsf{in}(c, y).\nu ch.(\mathsf{out}(ch, y).P_1^{ch} \mid !\mathsf{in}(ch, x)) \\
&\approx_\ell \mathsf{in}(c, y).\nu ch.(P_1^{ch} \mid !\mathsf{in}(ch, x)) \\
&\widehat{=} \ \mathsf{in}(c, y).P_1^{ch\backslash out(ch,.)} \\
&\approx_\ell \mathsf{in}(c, y).P_1
\end{aligned}$$

  To establish the last step, we can see that for any ground term M, the processes Q_1 and Q_2 such that $\mathsf{in}(c, y).P_1^{ch\backslash out(ch,.)} \xrightarrow{in(c,M)} Q_1$ and $\mathsf{in}(c, y).P_1 \xrightarrow{in(c,M)} Q_2$ are such that $Q_1 \equiv P_1\{^M/_y\}^{ch\backslash out(ch,.)}$ and $Q_2 \equiv P_1\{^M/_y\}$. By induction hypothesis, we have that Q_1 and Q_2 are bisimilar. Note that for this step we assume that w.l.o.g. $ch \notin fv(M)$. This can always be obtained by α-renaming ch. Lastly, we conclude thanks to the fact that $\mathsf{in}(c, y).P_1 = P$.
- $P = \mathsf{out}(c, M).P_1$. Note that $c \neq ch$. We have:

$$\begin{aligned}
P^{ch\backslash out(ch,.)} &= (\mathsf{out}(c, M).P_1)^{ch\backslash out(ch,.)} \\
&\widehat{=} \ \nu ch.(\mathsf{out}(c, M).P_1^{ch} \mid !\mathsf{in}(ch, x)) \\
&\approx_\ell \mathsf{out}(c, M).\nu ch.(P_1^{ch} \mid !\mathsf{in}(ch, x)) \\
&\widehat{=} \ \mathsf{out}(c, M).P_1^{ch\backslash out(ch,.)} \\
&\approx_\ell \mathsf{out}(c, M).P_1 \qquad \text{by induction hypothesis} \\
&= \ P
\end{aligned}$$

- $P = !P_1$. In such a case, we have:

$$\begin{aligned}
P^{ch\backslash out(ch,.)} &\widehat{=} \ (!P_1)^{ch\backslash out(ch,.)} \\
&\widehat{=} \ \nu ch.(!P_1^{ch} \mid !\mathsf{in}(ch, x)) \\
&\approx_\ell \nu ch.!(P_1^{ch} \mid !\mathsf{in}(ch, x)) \\
&\approx_\ell !(\nu ch.(P_1^{ch} \mid !\mathsf{in}(ch, x))) \ \text{ since } \mathsf{in}(ch, .) \text{ does not occur in } P_1^{ch} \\
&\widehat{=} \ !P_1^{ch\backslash out(ch,.)} \\
&\approx_\ell !P_1 \qquad \text{by induction hypothesis} \\
&= \ P
\end{aligned}$$

- $P =$ if $M = N$ then P_1 else P_2. Hence, we have:

$$
\begin{aligned}
P^{ch\backslash out(ch,.)} &= (\text{if } M = N \text{ then } P_1 \text{ else } P_2)^{ch\backslash out(ch,.)} \\
&\widehat{=} \ \nu ch.(\text{if } M = N \text{ then } P_1^{ch} \text{ else } P_2^{ch} \mid !\mathsf{in}(ch, x)) \\
&\approx_\ell \nu ch.(\text{if } M = N \text{ then } (P_1^{ch} \mid !\mathsf{in}(ch, x) \text{ else } (P_2^{ch} \mid !\mathsf{in}(ch, x))) \\
&\approx_\ell \nu ch.(\text{if } M = N \text{ then } (P_1^{ch} \mid !\mathsf{in}(ch, x) \text{ else } (P_2^{ch} \mid !\mathsf{in}(ch, x))) \\
&\approx_\ell \text{if } M = N \text{ then } \nu ch.(P_1^{ch} \mid !\mathsf{in}(ch, x)) \text{ else } \nu ch.(P_2^{ch} \mid !\mathsf{in}(ch, x)) \\
&\qquad\qquad \text{since } \mathsf{in}(ch, .) \text{ occurs neither in } P_1^{ch} \text{ nor in } P_2^{ch} \\
&\widehat{=} \ \text{if } M = N \text{ then } P_1^{ch\backslash out(ch,.)} \text{ else } P_2^{ch\backslash out(ch,.)} \\
&\approx_\ell \text{if } M = N \text{ then } P_1 \text{ else } P_2 \\
&= \ P
\end{aligned}
$$

This last case conludes the proof. □

Election verifiability in electronic voting protocols*

Steve Kremer[1], Mark Ryan[2], and Ben Smyth[2,3]

[1]LSV, ENS Cachan & CNRS & INRIA, France
[2]School of Computer Science, University of Birmingham, UK
[3]École Normale Supérieure, CNRS, INRIA, Paris, France

Technical Report CSR-10-06

April 9, 2010
(Revised: June 28, 2010)

Abstract

We present a symbolic definition of election verifiability for electronic voting protocols in the context of the applied pi calculus. Our definition is given in terms of boolean tests which can be performed on the data produced by an election. The definition distinguishes three aspects of verifiability, which we call individual verifiability, universal verifiability, and eligibility verifiability. It also allows us to determine precisely which aspects of the system's hardware and software must be trusted for the purpose of election verifiability. In contrast with earlier work our definition is compatible with a large class of electronic voting schemes, including those based on blind signatures, homomorphic encryption and mixnets. We demonstrate the applicability of our formalism by analysing two protocols which have been deployed; namely Helios 2.0, which is based on homomorphic encryption, and Civitas, which uses mixnets. In addition we consider the FOO protocol which is based on blind signatures.

1 Introduction

Electronic voting systems are being introduced, or trialled, in several countries to provide more efficient voting procedures with an increased level of security.

*This work has been partly supported by the EPSRC projects *UbiVal* (EP/D076625/2), *Trustworthy Voting Systems* (EP/G02684X/1) and *Verifying Interoperability Requirements in Pervasive Systems* (EP/F033540/1); the *ANR SeSur AVOTÉ* project; and the *Direction Générale pour l'Armement* (DGA).

However, the security of electronic elections has been seriously questioned [9, 19, 8, 23]. A major difference with traditional paper based elections is the lack of transparency. In paper elections it is often possible to observe the whole process from ballot casting to tallying, and to rely on robustness characteristics of the physical world (such as the impossibility of altering the markings on a paper ballot sealed inside a locked ballot box). By comparison, it is not possible to observe the electronic operations performed on data. Moreover, computer systems may alter voting records in a way that cannot be detected by either voters or election observers. For example, a voting terminal's software might be infected by malware which could change the vote entered by the user, or even execute a completely different protocol than the one expected. The situation can be described as *voting on Satan's computer*, analogously with [5]. Computer systems and election administrators should therefore be considered to be part of the adversary model.

The concept of *election verifiability* that has emerged in the academic literature, for example, [17, 18, 10, 3], aims to address this problem. It should allow voters and election observers to verify independently that votes have been recorded, tallied and declared correctly. To emphasise a voter's ability to verify the results of the entire election process, it is sometimes called *end-to-end* verifiability [20, 2]. The verification is performed using hardware and software of the verifier's own choice, and is completely independent of the hardware and software running the election. One generally distinguishes two aspects of verifiability.

- *Individual verifiability:* a voter can check that her own ballot is included in the bulletin board.
- *Universal verifiability:* anyone can check that the election outcome corresponds to the ballots published on the bulletin board.

We identify another aspect of verifiability which is sometimes included in universal verifiability.

- *Eligibility verifiability:* anyone can check that each vote in the election outcome was cast by a registered voter and there is at most one vote per voter.

We explicitly distinguish eligibility verifiability as a distinct property for compatibility with a larger class of protocols.

In this paper we present a symbolic definition of election verifiability for electronic voting protocols which captures the three desirable aspects. We model voting protocols in the applied pi calculus and formalise the different aspects of verifiability as a triple of boolean tests $\Phi^{IV}, \Phi^{UV}, \Phi^{EV}$. The test Φ^{IV} is intended to be checked by the individual voter who instantiates the test with her private information (for example, her vote and data derived during the execution of the protocol) and the public information available on the bulletin board. The tests Φ^{UV} and Φ^{EV} can be checked by any external observer and only rely on public information, that is, the contents of the bulletin board which

may include, for example, the set of ballots cast by voters, the list of eligible voters and the declared outcome. Our definition requires that these tests satisfy several conditions on all possible executions of the protocol. The consideration of eligibility verifiability is particularly interesting because it is essential to provide an assurance that the election outcome corresponds to votes legitimately cast and hence provides a mechanism to detect ballot stuffing.

A further interesting aspect of our work is the clear identification of which parts of the voting system need to be trusted to achieve verifiability. As already discussed it is not reasonable to assume voting systems behave correctly. Accordingly, when modelling a voting protocol as a process, we only model the parts of the protocol that we need to trust for the purpose of verifiability; all the remaining parts of the system will be controlled by the adversarial environment. Ideally, such a process would only model the interaction between a *voter* and the voting terminal; *that is, the messages input by the voter.* In particular, the voter should not need to trust the election hardware or software. However, achieving absolute verifiability in this context is difficult and we sometimes need to make explicit trust assumptions about which parts of the voter and administrator processes need to be trusted. As an example, when showing that the protocol by Fujioka *et al.* [15] ensures individual and universal verifiability we model the protocol as $\nu r.\overline{c}\langle v\rangle.\overline{c}\langle r\rangle$: the voter needs to generate a fresh nonce r and then give her vote v and r to the voting terminal, which is part of the adversarial environment. When the protocol is executed correctly this nonce is used to compute a commitment to the vote. This can be checked by the tests that ensure verifiability. The fact that νr is part of the protocol model implies that the nonce needs to be fresh for verifiability to hold. Hence, in this example the voter either needs to have a means to provide a fresh nonce or trust some part of the process to generate it freshly. Such trust assumptions are motivated by the fact that parts of a protocol can be audited, or because they can be executed in a distributed manner amongst several different election officials. For example, in the Helios 2.0 voting protocol [3], ballot construction can be audited using a cast-or-audit mechanism. Since any third party software can be used to audit the ballots the voters are assured that the ballots cast were constructed according to the protocol specification with high probability. Whether these trust assumptions are reasonable depends on the context of the given election.

We also note that the tests Φ^{IV}, Φ^{UV} and Φ^{EV} are assumed to be verified in a trusted environment. Indeed, if a test is checked by malicious software that always evaluates the test to hold, it is not of great value. However, the verification of these tests, unlike the election, can be repeated sufficiently many times, on different machines and using different software, which could be provided by different stakeholders of the election. Another possibility to avoid this issue would be to have tests which are human-verifiable as discussed for instance in [2, Chapter 5].

We demonstrate the applicability of our definition with three case studies: the protocol by Fujioka, Okamoto and Ohta [15]; the Helios 2.0 protocol [4] which was effectively used in recent university elections in Belgium; and the protocol by Juels, Catalano and Jakobsson [18], which has been implemented

by Clarkson, Chong and Myers as Civitas [13, 12]. Among other properties we show that the Helios protocol does not guarantee eligibility verifiability and is therefore vulnerable to ballot stuffing by dishonest administrators. As the protocol description does not mandate this property we do not claim this to be an attack, but simply clarify which aspects of verifiability are satisfied.

1.1 Our contribution

Our contribution is as follows:

1. A symbolic definition of election verifiability that considers a large class of protocols; including schemes based on: mixnets, homomorphic encryption and blind signatures. (In contrast, our preliminary work presented in [21] only considers blind signature schemes.)

2. Sound and intuitive consideration for eligibility verifiability. (A property which has been largely neglected and which our earlier work [21] provided only limited scope for.)

3. Formal treatment of trust assumptions for the purpose of verifiability.

In addition, the applicability of our work is demonstrated with respect to three case studies; namely, Helios 2.0, Civitas and FOO. The consideration of Helios 2.0 and Civitas is of particular interest since these systems have been implemented and deployed.

1.2 Related work

Juels *et al.* [17, 18] present a definition of universal verifiability in the provable security model. Their definition assumes voting protocols produce non-interactive zero-knowledge proofs of knowledge demonstrating the correctness of tallying. Here we consider definitions in a symbolic model. Universal verifiability was also studied by Chevallier-Mames *et al.* [11] with the aim of showing an incompatibility result: protocols cannot satisfy verifiability and vote privacy in an unconditional way (without relying on computational assumptions). But as witnessed by [17, 18], weaker versions of these properties can hold simultaneously. Our case studies demonstrate that our definition allows privacy and verifiability to coexist (see [14, 6] for a study of privacy properties in the applied pi calculus). Baskar *et al.* [7] and subsequently Talbi *et al.* [22] have formalised individual and universal verifiability with respect to the protocol by Fujioka *et al.* [15]. Their definitions are tightly coupled to that particular protocol and cannot easily be generalised. Moreover, their definitions characterise individual executions as verifiable or not; whereas such properties should be considered with respect to every execution (that is, the entire protocol).

In our earlier work [21] a preliminary definition of election verifiability was presented with support for automated reasoning. However, that definition is too strong to hold on protocols such as [18, 4]. In particular, our earlier definition

was only illustrated on a simplified version of [18] which did not satisfy privacy because we omitted the mixnets. Hence, this is the first general, symbolic definbition which can be used to show verifiability for many important protocols, such as the ones studied in this paper.

2 Applied pi calculus

The applied pi calculus [1, **?**] is a language for modelling concurrent, communicating processes. It is an extension of the pi calculus which was explicitly designed for modelling cryptographic protocols. For this purpose, the applied pi calculus allows processes to send terms constructed over a signature rather than just names. This term algebra can be used to model cryptographic primitives.

2.1 Syntax

The calculus assumes an infinite set of names $a, b, c, k, m, n, s, t, \ldots$, an infinite set of variables $v, x, y, z, \ldots$ and a finite signature Σ, that is, a finite set of function symbols each with an associated arity. A function symbol of arity 0 is a constant. We use metavariables u, w to range over both names and variables. Terms L, M, N, T, U, V are built by applying function symbols to names, variables and other terms. Tuples $u_1, \ldots, u_l$ and $M_1, \ldots, M_l$ are occasionally abbreviated $\tilde{u}$ and $\tilde{M}$. We write $\{^{M_1}/_{x_1}, \ldots, ^{M_l}/_{x_l}\}$ for substitutions that replace variables $x_1, \ldots, x_l$ with terms $M_1, \ldots, M_l$.

The applied pi calculus relies on a simple sort system. Terms can be of sort Channel for channel names or Base for the payload sent out on these channels. In addition we assume an infinite set of *record variables*. Function symbols can only be applied to, and return, terms of sort Base. A term is ground when it does not contain variables.

The grammar for processes is shown in Figure 1 where u is either a name or variable of channel sort. Plain processes are standard constructs, except for the *record message* $\mathsf{rec}(r, M).P$ construct which we discuss below. Extended processes introduce *active substitutions* which generalise the classical let construct: the process $\nu\, x.(\{^{M}/_{x}\} \mid P)$ corresponds exactly to the process let $x = M$ in P. As usual names and variables have scopes which are delimited by restrictions and by inputs. All substitutions are assumed to be cycle-free.

A *frame* φ is an extended process built from 0 and active substitutions $\{^{M}/_{x}\}$; which are composed by parallel composition and restriction. The *domain* of a frame φ is the set of variables that φ exports. Every extended process A can be mapped to a frame $\phi(A)$ by replacing every plain process in A with 0.

The record message construct $\mathsf{rec}(r, M).P$ introduces the possibility to enter special entries in frames. We suppose that the sort system ensures that r is a variable of record sort, which may only be used as a first argument of the rec construct or in the domain of the frame. Moreover, we make the global assumption that a record variable has a unique occurrence in each process.

Figure 1 Applied pi calculus grammar

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

Intuitively, this construct will be used to allow a voter to privately record some information which she may later use to verify the election; for example, nonces constructed during an execution of the protocol and/or messages received as input.

The sets of free and bound names, respectively variables, in process A are denoted by $\mathrm{fn}(A)$, $\mathrm{bn}(A)$, $\mathrm{fv}(A)$, $\mathrm{bv}(A)$. We also write $\mathrm{fn}(M)$, $\mathrm{fv}(M)$ for the names, respectively variables, in term M. Similarly, we write $\mathrm{rv}(A)$ and $\mathrm{rv}(M)$ for the set of record variables in a process, respectively a term. An extended process A is *closed* if it has no free variables. A *context* $C[\_]$ is an extended process with a hole. We obtain $C[A]$ as the result of filling $C[\_]$'s hole with A. An *evaluation context* is a context whose hole is not under a replication, a conditional, an input, or an output.

The signature Σ is equipped with an equational theory E, that is, a finite set of equations of the form $M = N$. We define $=_E$ as the smallest equivalence relation on terms, that contains E and is closed under application of function symbols, substitution of terms for variables and bijective renaming of names.

Example 1. *Let* $\Sigma = \{\mathsf{pair}(\cdot,\cdot), \mathsf{fst}(\cdot), \mathsf{snd}(\cdot)\}$ *and* E *be defined over the equations*

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

That is, the theory that models pairing and projection. Hence we have that $\mathsf{fst}(\mathsf{snd}(\mathsf{pair}(a, \mathsf{pair}(b, c)))) =_E b$.

In this paper we tacitly assume that all signatures and equational theories contain the function symbols $\mathsf{pair}(\cdot,\cdot), \mathsf{fst}(\cdot), \mathsf{snd}(\cdot)$ and equations for pairing as well as some constant $\bot$. As a convenient shortcut we then write $(\!|T_1, \ldots T_n|\!)$ for $\mathsf{pair}(T_1, \mathsf{pair}(\ldots, \mathsf{pair}(T_n, \bot)))$ and $\pi_i(T)$ for $\mathsf{fst}(\mathsf{snd}^{i-1}(T))$.

2.2 Semantics

We now define the operational semantics of the applied pi calculus by the means of three relations: structural equivalence, internal reductions and labelled reduction.

Structural equivalence ($\equiv$) is the smallest equivalence relation closed under α-conversion of both bound names and variables and application of evaluation contexts such that:

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

Internal reduction ($\rightarrow$) is the smallest relation closed under structural equivalence, application of evaluation contexts and such that:

$$
\begin{array}{ll}
\textsc{Rec} & \mathsf{rec}(r, M).P \rightarrow P \mid \{^M/_r\} \\
\textsc{Comm} & \overline{c}\langle x\rangle.P \mid c(x).Q \rightarrow P \mid Q \\
\textsc{Then} & \text{if } N = N \text{ then } P \text{ else } Q \rightarrow P \\
\textsc{Else} & \text{if } L = M \text{ then } P \text{ else } Q \rightarrow Q \\
 & \quad \text{for ground terms } L, M \text{ where } L \neq_E M
\end{array}
$$

Labelled reduction ($\xrightarrow{\alpha}$) extends internal reduction and enables the environment to interact with the processes using the rules defined below. The label α is either an input, or the output of a channel name or a variable of base sort.

$$a(x).P \xrightarrow{a(M)} P\{^M/_x\} \qquad \mathrm{rv}(M) = \emptyset$$

$$\overline{a}\langle u\rangle.P \xrightarrow{\overline{a}\langle u\rangle} P$$

$$\frac{A \xrightarrow{\overline{a}\langle u\rangle} A' \qquad u \neq a}{\nu u.A \xrightarrow{\nu u.\overline{a}\langle u\rangle} A'}$$

$$\frac{A \xrightarrow{\alpha} A' \qquad u \text{ does not occur in } \alpha}{\nu u.A \xrightarrow{\alpha} \nu u.A'}$$

$$\frac{A \xrightarrow{\alpha} A' \quad bv(\alpha) \cap fv(B) = bn(\alpha) \cap fn(B) = \emptyset}{A \mid B \xrightarrow{\alpha} A' \mid B}$$

$$\frac{A \equiv B \quad B \xrightarrow{\alpha} B' \quad A' \equiv B'}{A \xrightarrow{\alpha} A'}$$

We write $\Longrightarrow$ for $(\rightarrow^* \xrightarrow{\alpha} \rightarrow^*)^*$, that is, the reflexive transitive closure of the labelled reduction. We will not discuss these semantics in detail but give an example illustrating them (Figure 2).

Figure 2 A sequence of reductions in the applied pi semantics

Let $P = \nu a, b.\mathsf{rec}(r, a).\overline{c}\langle(\!|a, b|\!)\rangle.c(x).\text{if } x = a \text{ then } \overline{c}\langle f(a)\rangle$. Then we have that

$$\begin{array}{rcl}
P & \rightarrow & \nu a, b.(\overline{c}\langle(\!|a, b|\!)\rangle.c(x).\text{if } x = a \text{ then } \overline{c}\langle f(a)\rangle \mid \{^{a}/_{r}\}) \\
& \equiv & \nu a, b.(\nu y_1.(\overline{c}\langle y\rangle.c(x).\text{if } x = a \text{ then } \overline{c}\langle f(a)\rangle \mid \{^{(\!|a,b|\!)}/_{y_1}\}) \mid \{^{a}/_{r}\}) \\
& \xrightarrow{\nu x.\overline{c}\langle x\rangle} & \nu a, b.(c(x).\text{if } x = a \text{ then } \overline{c}\langle f(a)\rangle \mid \{^{(\!|a,b|\!)}/_{y_1}\} \mid \{^{a}/_{r}\}) \\
& \xrightarrow{\nu x.c(\pi_1(y))} & \nu a, b.(\text{if } a = a \text{ then } \overline{c}\langle f(a)\rangle \mid \{^{(\!|a,b|\!)}/_{y_1}\} \mid \{^{a}/_{r}\}) \\
& \rightarrow & \nu a, b.(\overline{c}\langle f(a)\rangle \mid \{\!| \{^{(\!|a,b|\!)}/_{y_1}\} \mid \{^{a}/_{r}\}) \\
& \xrightarrow{\nu y_2.\overline{c}\langle y_2\rangle} & \nu a, b.(\text{if } a = a \text{ then } \overline{c}\langle f(a)\rangle \mid \{^{(\!|a,b|\!)}/_{y_1}\} \mid \{^{f(a)}/_{y_2}\} \mid \{^{a}/_{r}\}
\end{array}$$

Observe that each labelled output is done by reference and extends the domain of the process's frame.

3 Formalising voting protocols

As discussed in the introduction we want to explicitly specify the parts of the election protocol which need to be trusted (that is, those parts of the system for which no verifiable proof of correct behaviour is provided). Formally the trusted parts of the voting protocol can be captured using a voting process specification.

Definition 1 (Voting process specification)**.** *A* voting process specification *is a tuple $\langle V, A\rangle$ where V is a plain process without replication and A is a closed evaluation context such that $fv(V) = \{v\}$ and $rv(V) = \emptyset$.*

Given a voting process specification $\langle V, A\rangle$, integer $n \in \mathbb{N}$, and names $s_1, \ldots, s_n$ we can build the voting process

$$\mathsf{VP}_n(s_1, \ldots, s_n) = A[V_1 \mid \cdots \mid V_n]$$

where $V_i = V\{^{s_i}/_{v}\}$. Intuitively, $\mathsf{VP}_n(s_1, \ldots, s_n)$ models the protocol with n voters casting votes for candidates $s_1, \ldots, s_n$. Note that the votes $s_1, \ldots, s_n$ are not required to be distinct (several voters may cast votes for the same candidate).

Example 2. *Consider the following simple* raising hands *protocol. Every voter simply outputs her signed vote. We suppose that a trusted administrator first distributes keying material and outputs a list of signed public keys corresponding to the public credentials of eligible voters. Signatures are modeled by the equations*

$$\mathsf{checksign}(\mathsf{pk}(x), \mathsf{sign}(x, y)) = \mathsf{true} \qquad \mathsf{getmsg}(\mathsf{sign}(x, y)) = y$$

The administrator generating and distributing keys via a private channel d is modelled by the following context.

$$A \mathrel{\widehat{=}} \nu d.\nu skA.(!\nu skv.\overline{d}\langle skv\rangle.\overline{c}\langle\mathsf{sign}(skA, \mathsf{pk}(skv))\rangle \mid \{^{\mathsf{pk}(skA)}/_{x_{pkA}}\} \mid \_)$$

The active substitution $\{^{\mathsf{pk}(skA)}/_{x_{pkA}}\}$ *models the fact that the administrator's public key is known, e.g. published on the election bulletin board. The voter, whom receives his private key and then outputs his signed vote is modelled by the process:*

$$V \mathrel{\widehat{=}} d(x_{skv}).\overline{c}\langle(\!|\mathsf{pk}(x_{skv}), \mathsf{sign}(x_{skv}, v)|\!)\rangle$$

We will prove that this protocol trivially satisfies individual and universal verifiability in Section 4; and eligibility verifiability in Section 5.

For the purposes of individual verifiability the voter may be reliant on some data derived during the execution of the protocol. We must therefore keep track of all such values. Definition 2 achieves this objective using the record message construct.

Definition 2. *Let* rv *be an infinite list of distinct* record variables*. We define the function* R *on a finite process* P *without replication as* $\mathsf{R}(P) = \mathsf{R}_{\mathsf{rv}}(P)$ *and, for all lists* rv*:*

$$\begin{array}{lcl}
\mathsf{R}_{rv}(0) & \mathrel{\widehat{=}} & 0 \\
\mathsf{R}_{rv}(P \mid Q) & \mathrel{\widehat{=}} & \mathsf{R}_{odd(rv)}(P) \mid \mathsf{R}_{even(rv)}(Q) \\
\mathsf{R}_{rv}(\nu\, n.P) & \mathrel{\widehat{=}} & \nu\, n.\mathsf{rec}(head(rv), n).\mathsf{R}_{tail(rv)}(P) \\
\mathsf{R}_{rv}(u(x).P) & \mathrel{\widehat{=}} & u(x).\mathsf{rec}(head(rv), x).\mathsf{R}_{tail(rv)}(P) \\
\mathsf{R}_{rv}(\overline{u}\langle M\rangle.P) & \mathrel{\widehat{=}} & \overline{u}\langle M\rangle.\mathsf{R}_{rv}(P) \\
\mathsf{R}_{rv}(\mathit{if}\ M = N\ \mathit{then}\ P\ \mathit{else}\ Q) & \mathrel{\widehat{=}} & \mathit{if}\ M = N\ \mathit{then}\ \mathsf{R}_{rv}(P)\ \mathit{else}\ \mathsf{R}_{rv}(Q)
\end{array}$$

where the functions head and tail are the usual ones for lists, and odd (resp. even) returns the list of elements in odd (resp. even) position.

In the above definition odd and even are used as a convenient way to split an infinite list into two infinite lists. A voting process can now be constructed such that the voter V records the values constructed and input during execution.

Definition 3. *Given a voting process specification* $\langle V, A\rangle$*, integer* $n \in \mathbb{N}$*, and names* $s_1, \ldots, s_n$ *, we build the augmented voting process*

$$\mathsf{VP}^{+}_{n}(s_1, \ldots, s_n) = A[V_1^{+} \mid \cdots \mid V_n^{+}]$$

where $V_i^{+} = \mathsf{R}(V)\{^{s_i}/_{v}\}\{^{r_i}/_{r} \mid r \in rv(\mathsf{R}(V))\}$.

For notational purposes, given a sequence of record variables $\tilde{r}$, we denote by $\tilde{r}_i$ the sequence of variables obtained by indexing each variable in $\tilde{r}$ with i. The process $\mathsf{VP}^{+}_{n}(s_1, \ldots, s_n)$ models the voting protocol for n voters casting votes $s_1, \ldots, s_n$, who privately record the data that may be needed for verification using record variables $\tilde{r}_i$.

4 Election verifiability

We formalize election verifiability using three tests Φ^{IV}, Φ^{UV}, Φ^{EV}. Formally, a test is built from conjunctions and disjunctions of *atomic tests* of the form

$(M =_E N)$ where M, N are terms. Tests may contain variables and will need to hold on frames arising from arbitrary protocol executions. The test Φ^{IV} has record variables which will be substituted by the records stored in the frame; and variables expected to correspond to the voter's ballot and other public information, which will be other variables in the domain of the frame. The tests Φ^{UV}, Φ^{EV} substitute only public information, that is, (plain) variables in the frame's domain and hence are suitable for the use by election observers. The designers of electronic voting protocols need not explicitly specify cryptographic tests Φ^{IV}, Φ^{UV}, Φ^{EV} since our definition assumes the existence of tests (perhaps devised after design) which satisfy our conditions. Now we recall the purpose of each test and assume some conventions about how variables are named in the tests.

Individual verifiability: The test Φ^{IV} allows a voter to identify her ballot in the bulletin board. The test has:

- a variable v referring to a voter's vote.
- a variable w referring to a voter's public credential.
- some variables $x, \bar{x}, \hat{x}, \ldots$ expected to refer to global public values pertaining to the election, for example, public keys belonging to election administrators.
- a variable y expected to refer to the voter's ballot on the bulletin board.
- some record variables $r_1, \ldots, r_k$ referring to the voter's private data.

Universal verifiability: The test Φ^{UV} allows an observer to check that the election outcome corresponds to the ballots in the bulletin board. The test has:

- a tuple of variables $\tilde{v} = (v_1, \ldots, v_n)$ referring to the declared outcome.
- some variables $x, \bar{x}, \hat{x}, \ldots$ as above.
- a tuple $\tilde{y} = (y_1, \ldots, y_n)$ expected to refer to all the voters' ballots on the bulletin board.
- some variables $z, \bar{z}, \hat{z}, \ldots$ expected to refer to outputs generated during the protocol used for the purposes of universal and eligibility verification.

Eligibility verifiability: The test Φ^{EV} allows an observer to check that each ballot in the bulletin board was cast by a unique registered voter. The test has:

- a tuple $\tilde{w} = (w_1, \ldots, w_n)$ referring to public credentials of eligible voters.
- some variables $x, \bar{x}, \hat{x}, \ldots$ as above.
- a tuple $\tilde{y}$ as above.
- some variables $z, \bar{z}, \hat{z}, \ldots$ as above.

The remainder of this section will focus on the individual and universal aspects of our definition; eligibility verifiability will be discussed in Section 5.

4.1 Individual and universal verifiability

The tests suitable for the purposes of election verifiability have to satisfy certain conditions: if the tests succeed, then the data output by the election is indeed valid (*soundness*); and there is a behaviour of the election authority which produces election data satisfying the tests (*effectiveness*). Formally these requirements are captured by the definition below. We use the notation $\tilde{T} \simeq \tilde{T}'$ to denote that the tuples $\tilde{T}$ and $\tilde{T}'$ are a permutation of each others modulo the equational theory, that is, we have $\tilde{T} = T_1, \ldots T_n$, $\tilde{T}' = T'_1, \ldots T'_n$ and there exists a permutation χ on $\{1, \ldots, n\}$ such that for all $1 \leq i \leq n$ we have $T_i =_E T'_{\chi(i)}$.

Definition 4 (Individual and universal verifiability)**.** *A voting specification* $\langle V, A\rangle$ *satisfies* individual and universal verifiability *if for all* $n \in \mathbb{N}$ *there exist tests* Φ^{IV}, Φ^{UV} *such that* $fn(\Phi^{IV}) = fn(\Phi^{UV}) = rv(\Phi^{UV}) = \emptyset$, $rv(\Phi^{IV}) \subseteq rv(\mathsf{R}(V))$*, and for all names* $\tilde{s} = (s_1, \ldots, s_n)$ *the conditions below hold. Let* $\tilde{r} = rv(\Phi^{IV})$ *and* $\Phi^{IV}_i = \Phi^{IV}\{^{s_i}/_v, ^{\tilde{r}_i}/_{\tilde{r}}\}$.

Soundness. *For all contexts* C *and processes* B *such that* $C[\mathsf{VP}^+_n(s_1, \ldots, s_n)] \Longrightarrow B$ *and* $\phi(B) \equiv \nu\tilde{n}.\sigma$*, we have:*

$$\forall i, j. \quad \Phi^{IV}_i\sigma \wedge \Phi^{IV}_j\sigma \Rightarrow i = j \tag{1}$$

$$\Phi^{UV}\sigma \wedge \Phi^{UV}\{^{\tilde{v}'}/_{\tilde{v}}\}\sigma \Rightarrow \tilde{v}\sigma \simeq \tilde{v}'\sigma \tag{2}$$

$$\bigwedge_{1\leq i\leq n} \Phi^{IV}_i\{^{y_i}/_y\}\sigma \wedge \Phi^{UV}\sigma \;\Rightarrow\; \tilde{s} \simeq \tilde{v}\sigma \tag{3}$$

Effectiveness. *There exists a context* C *and a process* B*, such that* $C[\mathsf{VP}^+_n(s_1, \ldots, s_n)] \Longrightarrow B$*,* $\phi(B) \equiv \nu\tilde{n}.\sigma$ *and*

$$\bigwedge_{1\leq i\leq n} \Phi^{IV}_i\{^{y_i}/_y\}\sigma \wedge \Phi^{UV}\sigma \tag{4}$$

We now discuss how voters and observers use these tests and what are the guarantees given by the conditions stated in Definition 4.

An individual voter should verify that the test Φ^{IV} holds when instantiated with her vote s_i, the information $\tilde{r}_i$ recorded during the execution of the protocol and some bulletin board entry (which she needs to identify in some way, maybe by testing all of them). Indeed, Condition (1) ensures that the test will hold for at most one bulletin board entry. This allows the voter to convince herself that her ballot has been counted. (To understand the way the condition is encoded, notice that in the first conjunct, the test succeeds with the ith voter's data and a ballot $y\sigma$ provided by the context $C[\_]$; in the second conjunct, the test succeeds with j's data and the same ballot.) The fact that her ballot contains her vote will be ensured by Φ^{UV} which should also be tested by the voter.

An observer will instantiate the test Φ^{UV} with the bulletin board entries $\tilde{y}$ and the declared outcome $\tilde{v}$. Condition (2) ensures the observer that Φ^{UV} only

holds for one outcome. (In the first and second conjuncts, the test succeeds with declared outcomes $\tilde{v}\sigma$ and $\tilde{v}'\sigma$ respectively, where both $\tilde{v}\sigma$ and $\tilde{v}'\sigma$ are provided by the context $C[\_]$.)

Condition (3) ensures that if a bulletin board contains the ballots of voters who voted $s_1, \ldots, s_n$ then Φ^{UV} only holds if the declared outcome is (a permutation of) these votes.

Finally, Condition (4) ensures that there exists an execution where the tests hold. In particular this allows us to verify whether the protocol can satisfy the tests when executed as expected. This also avoids tests which are always false and would make Conditions (1)-(3) vacuously hold.

Example 3. *We show that the raising hands protocol (Example 2) satisfies our definition. Note that in the augmented voting process, the voter will record his private key; we will denote the ith voter's private key with the record variable r_{skv_i}. For all $n \in \mathbb{N}$ we define the tests*

$$\Phi^{IV} \mathrel{\hat{=}} y =_E (\!|\mathsf{pk}(r_{skv}), \mathsf{sign}(r_{skv}, v)|\!) \qquad \Phi^{UV} \mathrel{\hat{=}} \bigwedge_{1 \le i \le n} \mathsf{getmsg}(\pi_2(y_i)) =_E v_i$$

We now show that Conditions (1)–(3) of Definition 4 are satisfied.

(1) Suppose that $\Phi_i^{IV}\sigma$ and $\Phi_j^{IV}\sigma$ hold, that is,

$$\begin{array}{lll} y\sigma & =_E & (\!|\mathsf{pk}(r_{skv_i}\sigma), \mathsf{sign}(r_{skv_i}\sigma, s_i)|\!) \\ y\sigma & =_E & (\!|\mathsf{pk}(r_{skv_j}\sigma), \mathsf{sign}(r_{skv_j}\sigma, s_j)|\!) \end{array}$$

From the equational theory it follows that $r_{skv_i}\sigma =_E r_{skv_j}\sigma$. Moreover, it follows from the voting process specification and the semantics of the applied pi calculus that for every σ, such that $C[\mathsf{VP}_n^+(s_1, \ldots, s_n)] \Longrightarrow B$ and $\phi(B) \equiv \nu\tilde{n}.\sigma$, $i \neq j$ implies that $r_{skv_i}\sigma \neq_E r_{skv_j}\sigma$. Hence we conclude that Condition (1) holds.

(2) For any substitution σ, the premise of Condition (2) implies $\bigwedge_{1 \le i \le n} v_i\sigma =_E v_i'\sigma$ and hence $v\sigma \simeq v'\sigma$.

(3) For any substitution σ, the premise of Condition (3) implies $\bigwedge_{1 \le i \le n} s_i =_E \pi_1(y_i\sigma) \wedge \pi_1(y_i\sigma) =_E v_i\sigma$ and hence $\tilde{s} \simeq \tilde{v}\sigma$.

To see that Condition (4) holds let $C \mathrel{\hat{=}} \_$. It is easy to see that $\mathsf{VP}_n^+(s_1, \ldots, s_n) \Longrightarrow B$, such that

$$\phi(B) \equiv \nu skA, skv_1 \ldots skv_n.\{^{\mathsf{pk}(skA)}/_{x_{pkA}}, ^{(\!|\mathsf{pk}(skv_1), \mathsf{sign}(skv_1, s_1)|\!)}/_{y_1}, ^{skv_1}/_{r_{skv_1}}, \ldots, ^{(\!|\mathsf{pk}(skv_n), \mathsf{sign}(skv_n, s_n)|\!)}/_{y_n}, ^{skv_n}/_{r_{skv_n}}\}$$

and that $\Phi^{IV}\sigma \wedge \Phi^{UV}\sigma$ hold.

Example 4. *Consider the* postal vote *protocol whereby all voters simply send their vote to an administrator who publishes the list of cast votes. The voting process specification is simply $\langle \overline{c}\langle v\rangle, \_\rangle$. Such a protocol is obviously not verifiable and violates our definition. It is indeed not possible to design a test Φ^{IV} such that Condition (1) holds when $s_i = s_j$ for some $i \neq j$.*

4.2 Case study: FOO

The protocol by Fujioka, Okamoto and Ohta [15], FOO for short, was an early protocol based on blind signatures and has been influential for the design of later protocols.

How FOO works. The FOO protocol involves voters, a registrar and a tallier. The voter first computes her ballot as a commitment to her vote $m' = \mathsf{commit}(rnd, v)$ and sends the signed blinded ballot $\mathsf{sign}(sk_V, \mathsf{blind}(rnd', m'))$ to the registrar. The registrar checks that the signature belongs to an eligible voter and returns $\mathsf{sign}(sk_R, \mathsf{blind}(rnd', m'))$ the blind signed ballot. The voter verifies that this input corresponds to the registrar's signature and unblinds the message to recover her ballot signed by the registrar $m = \mathsf{sign}(sk_R, m')$. The voter then posts her signed ballot to the bulletin board. Once all votes have been cast the tallier verifies all the entries and appends an identifier l to each valid entry. The voter then checks the bulletin board for her entry, the triple (l, m', m), and appends the commitment factor rnd. Finally, using rnd the tallier opens all of the ballots and announces the declared outcome.

Equational theory. We model blind signatures and commitment as follows.

$$\begin{array}{rclrcl} \mathsf{checksign}(\mathsf{pk}(x), \mathsf{sign}(x,y)) & = & \mathsf{true} & \mathsf{getmsg}(\mathsf{sign}(x,y)) & = & y \\ \mathsf{unblind}(y, \mathsf{sign}(x, \mathsf{blind}(y,z))) & = & \mathsf{sign}(x,z) & \mathsf{unblind}(x, \mathsf{blind}(x,y)) & = & y \\ \mathsf{open}(x, \mathsf{commit}(x,y)) & = & y & & & \end{array}$$

Model in applied pi. As discussed in the introduction, the parts of the protocol that need to be trusted for achieving verifiability are surprisingly simple. The name rnd models the randomness that is supposed to be used to compute the commitment of the vote. All a voter needs to ensure is that the randomness used for the commitment is fresh. To ensure verifiability, all other operations such as computing the commitment, blinding and signing can be performed by the untrusted terminal.

Definition 5. *The voting process specification* $\langle V_{\mathsf{foo}}, A_{\mathsf{foo}} \rangle$ *is defined where*

$$V_{\mathsf{foo}} \mathrel{\hat{=}} \nu rnd.\overline{c}\langle v \rangle.\overline{c}\langle rnd \rangle \quad \textit{and} \quad A_{\mathsf{foo}}[\_] \mathrel{\hat{=}} \_$$

The name rnd models the randomness that is supposed to be used to compute the commitment of the vote. All a voter needs to ensure is that the randomness used for the commitment is fresh. To ensure verifiability, all other operations such as computing the commitment, blinding and signing can be performed by the untrusted terminal. The augmented voting process $\mathsf{VP}_n^+(s_1, \ldots, s_n)$ is $\nu\ rnd.\mathsf{rec}(r_1, rnd).\overline{c}\langle s_1 \rangle.\overline{c}\langle rnd \rangle \mid \ldots \mid \nu\ rnd.\mathsf{rec}(r_n, rnd).\overline{c}\langle s_n \rangle.\overline{c}\langle rnd \rangle$.

Individual and universal verifiability. We define the tests

$$\Phi^{IV} \mathrel{\hat{=}} y =_E (\!|r, \mathsf{commit}(r,v)|\!) \qquad \Phi^{UV} \mathrel{\hat{=}} \bigwedge_{1 \le i \le n} v_i =_E \mathsf{open}(\pi_1(y), \pi_2(y))$$

Intuitively, a bulletin board entry y should correspond to the pair formed of the random generated by voter i and commitment to her vote.

Theorem 1. *$\langle V_{\mathsf{foo}}, A_{\mathsf{foo}}\rangle$ satisfies individual and universal verifiability.*

Proof. We show that the Conditions (1)–(3) of Definition 4 hold.

(1) Suppose C, B, i, j are such that $C[\mathsf{VP}^{+}_{\mathsf{n}}(s_1,\ldots,s_n)] \Longrightarrow B$, $\phi(B) \equiv \nu\tilde{n}.\sigma$, $\Phi_i^{IV}\sigma$ and $\Phi_j^{IV}\sigma$. Then $\pi_2(y)\sigma = r_i\sigma$ by $\Phi_i^{IV}\sigma$, and $\pi_2(y)\sigma = r_j\sigma$ by $\Phi_j^{IV}\sigma$, so $r_i\sigma = r_j\sigma$. But since these are randoms freshly generated by the processes V_i and V_j, it follows that $i = j$. (This can be easily shown by induction on the derivation which produces B.)

(2) For any σ we have for all $1 \le i \le n$ that

$$v_i\sigma =_E \mathsf{open}(\pi_1(y_i)\sigma, \pi_2(y_i)\sigma) \wedge v_i'\sigma =_E \mathsf{open}(\pi_1(y_i\sigma), \pi_2(y_i\sigma)) \Rightarrow v_i\sigma =_E v_i'\sigma$$

(3) It follows from the equational theory that for all $1 \le i \le n$ and substitution σ that

$$y_i\sigma =_E (\!|r_i\sigma, \mathsf{commit}(r_i\sigma, s_i)|\!) \wedge v_i\sigma =_E \mathsf{open}(\pi_1(y_i\sigma), \pi_2(y_i\sigma)) \Rightarrow \tilde{s} =_E \tilde{v}\sigma$$

It is also easy to see that a context modelling the entire FOO protocol would satisfy effectiveness (Condition (4)). One may for instance slightly adapt the modelling of the FOO protocol given in [14] for this purpose. □

Our model of FOO does not rely on the blind signatures. While this part is crucial for privacy properties it does not contribute to verifiability. Similarly, the voter's signature on the blinded committed vote and the confidentiality of the secret signing key are not required for individual and universal verifiability; they are however essential for eligibility.

4.3 Case study: Helios 2.0

Helios 2.0 [4] is an open-source web-based election system, based on homomorphic tallying of encrypted votes. It allows the secret election key to be distributed amongst several trustees, and supports distributed decryption of the election result. It also allows independent verification by voters and observers of election results. Helios 2.0 was successfully used in March 2009 to elect the president of the Catholic University of Louvain, an election that had 25,000 eligible voters.

How Helios works. An election is created by naming a set of trustees and running a protocol that provides each of them with a share of the secret part of a public key pair. The public part of the key is published. Each of the eligible voters is also provided with a private pseudo-identity. The steps that participants take during a run of Helios are as follows.

1. To cast a vote, the user runs a browser script that inputs her vote and creates a ballot that is encrypted with the public key of the election. The ballot includes a ZKP that the ballot represents an allowed vote (this is needed because the ballots are never decrypted individually).

2. The user can audit the ballot to check if it really represents a vote for her chosen candidate; if she elects to do this, the script provides her with the random data used in the ballot creation. She can then independently verify that the ballot was correctly constructed, but the ballot is now invalid and she has to create another one.

3. When the voter has decided to cast her ballot, the voter's browser submits it along with her pseudo-identity to the server. The server checks the ZKPs of the ballots, and publishes them on a bulletin board.

4. Individual voters can check that their ballots appear on the bulletin board. Any observer can check that the ballots that appear on the bulletin board represent allowed votes, by checking the ZKPs.

5. The server homomorphically combines the ballots, and publishes the encrypted tally. Anyone can check that this tally is done correctly.

6. The server submits the encrypted tally to each of the trustees, and obtains their share of the decryption key for that particular ciphertext, together with a proof that the key share is well-formed. The server publishes these key shares along with the proofs. Anyone can check the proofs.

7. The server decrypts the tally and publishes the result. Anyone can check this decryption.

Equational theory. We use a signature in which $\mathsf{penc}(x_{\mathsf{pk}}, x_{\mathsf{rand}}, x_{\mathsf{text}})$ denotes the encryption with key x_{pk} and random x_{rand} of the plaintext x_{text}, and $x_{\mathsf{ciph}} * y_{\mathsf{ciph}}$ denotes the homomorphic combination of ciphertexts x_{ciph} and y_{ciph} (the corresponding operation on plaintexts is written $+$ and on randoms $\circ$). The term $\mathsf{ballotPf}(x_{\mathsf{pk}}, x_{\mathsf{rand}}, s, x_{\mathsf{ballot}})$ represents a proof that the ballot x_{ballot} contains some name s and random x_{rand} with respect to key x_{pk}; $\mathsf{decKey}(x_{\mathsf{sk}}, x_{\mathsf{ciph}})$ is a decryption key for x_{ciph} w.r.t. public key $\mathsf{pk}(x_{\mathsf{sk}})$; and $\mathsf{decKeyPf}(x_{\mathsf{sk}}, x_{\mathsf{ciph}}, x_{\mathsf{dk}})$ is a proof that x_{dk} is a decryption key for x_{ciph} w.r.t. public key $\mathsf{pk}(x_{\mathsf{sk}})$. We use the equational theory that asserts that $+, *, \circ$ are commutative and associative,

and includes the equations:

$$\mathsf{dec}(x_{\mathsf{sk}}, \mathsf{penc}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{rand}}, x_{\mathsf{text}})) = x_{\mathsf{text}}$$

$$\mathsf{dec}(\mathsf{decKey}(x_{\mathsf{sk}}, ciph), ciph) = x_{\mathsf{plain}}$$
where $ciph = \mathsf{penc}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{rand}}, x_{\mathsf{plain}})$

$$\mathsf{penc}(x_{\mathsf{pk}}, y_{\mathsf{rand}}, y_{\mathsf{text}}) * \mathsf{penc}(x_{\mathsf{pk}}, z_{\mathsf{rand}}, z_{\mathsf{text}}) = \mathsf{penc}(x_{\mathsf{pk}}, y_{\mathsf{rand}} \circ z_{\mathsf{rand}}, y_{\mathsf{text}} + z_{\mathsf{text}})$$

$$\mathsf{checkBallotPf}(x_{\mathsf{pk}}, ballot, \mathsf{ballotPf}(x_{\mathsf{pk}}, x_{\mathsf{rand}}, s, ballot)) = \mathsf{true}$$
where $ballot = \mathsf{penc}(x_{\mathsf{pk}}, x_{\mathsf{rand}}, s)$

$$\mathsf{checkDecKeyPf}(\mathsf{pk}(x_{\mathsf{sk}}), ciph, dk, \mathsf{decKeyPf}(x_{\mathsf{sk}}, ciph, dk)) = \mathsf{true}$$
where $ciph = \mathsf{penc}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{rand}}, x_{\mathsf{plain}})$and $dk = \mathsf{decKey}(x_{\mathsf{sk}}, ciph)$

Note that in the equation for checkBallotPf we have that s is a name and not a variable. As the equational theory is closed under bijective renaming of names this equation will hold for any name, but will fail if one replaces the name by a term, for example, $s + s$. We suppose that all names are possible votes but give the possibility to check that a voter does not include a term $s + s$ which would allow her to add an additional vote to the outcome.

Model in applied pi. The parts of the system that are not verifiable are:

- The browser script that constructs the ballot. Although the voter cannot verify it, the trust in this script is motivated by the fact that she is able to audit it. She does that by creating as many ballots as she likes and checking all but one of them, and then casting the one she didn't verify.
- The trustees. Although the trustees' behaviour cannot be verified, voters and observers may want to trust them because trust is distributed among them.

We model these two components as trusted parts, by including them in the context A_{helios} of our voting process specification.

Definition 6. *The voting process specification* $\langle V_{\mathsf{helios}}, A_{\mathsf{helios}}\rangle$ *is defined where*

$$\begin{aligned}
V_{\mathsf{helios}} &\hat{=} d(x_{\mathsf{pid}}).\ \overline{d}\langle v\rangle.\ d(x_{\mathsf{ballot}}).\ d(x_{\mathsf{ballotpf}}).\overline{c}\langle(\!|w, x_{\mathsf{ballot}}, x_{\mathsf{ballotpf}}|\!)\rangle\\
A_{\mathsf{helios}}[_] &\hat{=} \nu sk, d.\ \left(\overline{c}\langle\mathsf{pk}(sk)\rangle \mid (!\nu pid.\ \overline{d}\langle pid\rangle) \mid (!B) \mid T \mid _\right)\\
B &\hat{=} \nu m.\ d(x_{\mathsf{vote}}).\overline{d}\langle\mathsf{penc}(\mathsf{pk}(sk), m, x_{\mathsf{vote}})\rangle.\\
&\qquad \overline{d}\langle\mathsf{ballotPf}(\mathsf{pk}(sk), m, x_{\mathsf{vote}}, \mathsf{penc}(\mathsf{pk}(sk), m, x_{\mathsf{vote}}))\rangle\\
T &\hat{=} c(x_{\mathsf{tally}}).\ \overline{c}\langle(\!|\mathsf{decKey}(sk, x_{\mathsf{tally}}), \mathsf{decKeyPf}(sk, x_{\mathsf{tally}}, \mathsf{decKey}(sk, x_{\mathsf{tally}}))|\!)\rangle
\end{aligned}$$

We suppose that the recording function records the inputs of x_{pid}, x_{ballot} and x_{ballotpf} in record variables r_{pid}, r_{ballot} and $r_{ballotpf}$ respectively. The voter V_{helios} receives her voter id pid on a private channel. She sends her vote on the channel to A_{helios}, which creates the ballot for her. She receives the ballot and sends it (paired with pid) to the server. A_{helios} represents the parts of the system that

are required to be trusted. It publishes the election key and issues voter ids. It includes the ballot creation script B, which receives a voter's vote, creates a random m and forms the ballot, along with its proof, and returns it to the voter. A_{helios} also contains the trustee T, which accepts a tally ciphertext and returns a decryption key for it, along with the proof that the decryption key is correct. We assume the trustee will decrypt any ciphertext (but only one). In practice, of course, the trustee should ensure that the ciphertext is the right one, namely, the homomorphic addition of all the ballots posted to the bulletin board.

The untrusted server is assumed to publish the election data. In our formalism, we expect the frame to have a substitution σ that defines the election public key as x_{pk} and the individual pid's and ballots as y_i for each voter i. It also contains the homomorphic tally z_{tally} of the encrypted ballots, and the decryption key z_{decKey} and its proof of correctness z_{decKeyPf} obtained from the trustees. When the protocol is executed as expected the resulting frame should have substitution σ such that

$$\begin{aligned}
x_{pk}\sigma &= \mathsf{pk}(sk)\\
y_i\sigma &= (\!|pid_i, \mathsf{penc}(\mathsf{pk}(sk), m_i, v_i),\\
&\qquad \mathsf{ballotPf}(\mathsf{pk}(sk), m_i, v_i, \mathsf{penc}(\mathsf{pk}(sk), m_i, v_i))|\!)\\
z_{\mathsf{tally}}\sigma &= \pi_2(y_1) * \cdots * \pi_2(y_n)\sigma\\
z_{\mathsf{decKey}}\sigma &= \mathsf{decKey}(sk, z_{\mathsf{tally}})\sigma\\
z_{\mathsf{decKeyPf}}\sigma &= \mathsf{decKeyPf}(sk, z_{\mathsf{tally}}, z_{\mathsf{decKey}})\sigma
\end{aligned}$$

The server then decrypts the tally to obtain the outcome of the election.

Individual and universal verifiability. For the purposes of individual and universal verifiability, the tests Φ^{IV} and Φ^{UV} are introduced. Accordingly, given $n \in \mathbb{N}$ we define:

$$\begin{aligned}
\Phi^{IV} &\mathrel{\widehat{=}} y =_E (\!|r_{pid}, r_{ballot}, r_{ballotpf}|\!)\\
\Phi^{UV} &\mathrel{\widehat{=}} z_{\mathsf{tally}} =_E \pi_2(y_1) * \cdots * \pi_2(y_n)\\
&\quad \wedge \textstyle\bigwedge_{i=1}^{n} (\mathsf{checkBallotPf}(x_{\mathsf{pk}}, \pi_2(y_i), \pi_3(y_i)) =_E \mathsf{true})\\
&\quad \wedge \mathsf{checkDecKeyPf}(x_{\mathsf{pk}}, z_{\mathsf{tally}}, z_{\mathsf{decKey}}, z_{\mathsf{decKeyPf}}) =_E \mathsf{true}\\
&\quad \wedge v_1 + \cdots + v_n =_E \mathsf{dec}(z_{\mathsf{decKey}}, z_{\mathsf{tally}})
\end{aligned}$$

The test Φ^{IV} checks that the voter's ballot is recorded on the bulletin board. The test Φ^{UV} checks that the tally is correctly computed; it checks the proof for the decryption key; and it checks the decrypted tally corresponds to the declared outcome $\tilde{v}$.

Theorem 2. *$\langle V_{\mathsf{helios}}, A_{\mathsf{helios}}\rangle$ satisfies individual and universal verifiability.*

Proof. Suppose $n \in \mathbb{N}$ and test Φ^{IV}, Φ^{UV} are given above. We will now show that for all $\tilde{s} = (s_1, \ldots, s_n)$ that the conditions of Definition 4 are satisfied.

(1) Suppose C, B, i, j are such that $C[\mathsf{VP}^{+}_{\mathsf{n}}(s_1, \ldots, s_n)] \Longrightarrow B$, $\phi(B) \equiv \nu\tilde{n}.\sigma$, and $\Phi^{IV}_i\sigma$ and $\Phi^{IV}_j\sigma$ hold. Then $\pi_2(y)\sigma = r_{ballot,i}\sigma$ by $\Phi^{IV}_i\sigma$, and

$\pi_2(y)\sigma = r_{ballot,j}\sigma$ by $\Phi_j^{IV}\sigma$, so $r_{ballot,i}\sigma = r_{ballot,j}\sigma$. But since these are randoms freshly generated for the processes V_i and V_j, it follows that $i = j$.

(2) Let σ be any substitution and suppose that $\Phi^{UV}\sigma$ and $\Phi^{UV}\{^{\tilde{v}'}/_{\tilde{v}}\}\sigma$. Then $(v_1 + \cdots + v_n)\sigma = (v'_1 + \cdots + v'_n)\sigma = \mathsf{dec}(z_{\mathsf{decKey}}, z_{\mathsf{tally}})\sigma$. Moreover, $\Phi^{UV}\sigma$ we have that $\bigwedge_{i=1}^{n}(\mathsf{checkBallotPf}(x_{\mathsf{pk}}, \pi_2(y_i), \pi_3(y_i)))\sigma$ which implies that each $v_i\sigma$ and $v'_i\sigma$ is a name. Hence $\tilde{v}\sigma \simeq \tilde{v}'\sigma$.

(3) Let σ be any substitution and suppose that $\bigwedge_{1\leq i\leq n}\Phi_i^{IV}\{^{y_i}/_{y}\}\sigma$ and $\Phi^{UV}\sigma$. From each $\Phi_i^{IV}\{^{y_i}/_{y}\}\sigma$, we have that $\pi_2(y_i)\sigma = \mathsf{penc}(\mathsf{pk}(sk), m_i, s_i)\sigma$ for some m_i. From $\Phi^{UV}\sigma$, we have $z_{\mathsf{tally}}\sigma = (\pi_2(y_1) * \cdots * \pi_2(y_n))\sigma$, and by the equation for homomorphic encryption, this is $\mathsf{penc}(\mathsf{pk}(sk), m_1 \circ \cdots \circ m_n, s_1 + \cdots + s_n)$. From the decryption key proof and the decryption, we have $(v_1 + \cdots + v_n)\sigma = (s_1 + \cdots + s_n)$, and from the ballot proofs, we can conclude that $\tilde{s} \simeq \tilde{v}\sigma$.

(4) The context C must marshal the election data on the frame in such a way that $x_{pk}\sigma$, $y_i\sigma$, $z_{\mathsf{tally}}\sigma$, $z_{\mathsf{decKey}}\sigma$, and $z_{\mathsf{decKeyPf}}\sigma$ are as defined above. Moreover, it finds some names $t_1, \ldots, t_n$ such that $z_{\mathsf{tally}}\sigma = t_1 + \cdots + t_n$, and sets the declared outcome $\tilde{v}\sigma$ to be $(\!|t_1, \ldots, t_n|\!)$. □

5 Eligibility verifiability

In order to fully capture *election verifiability*, the tests Φ^{IV} and Φ^{UV} must be supplemented by a test Φ^{EV} that checks eligibility of the voters whose votes have been counted in the outcome. We suppose that the public voter credentials appear on the bulletin board. Moreover, these credentials actually belong to eligible voters; verifying this is beyond the scope of this paper. One approach may involve publishing the list of credentials alongside the real names and addresses of the electorate, the validity of this list can then be scrutinised by the observer. The test Φ^{EV} allows an observer to check that only these individuals (that is, those in posession of credentials) cast votes, and at most one vote each. The test is instantiated with the list of public credentials, and other public outputs of the election process, such as the public keys, the voters' ballots, and any other outputs such as proofs. We use the variable naming convention introduced in the previous section.

Definition 7 (Election verifiability)**.** *A voting specification $\langle V, A\rangle$ satisfies* election verifiability *if for all $n \in \mathbb{N}$ there exist tests $\Phi^{IV}, \Phi^{UV}, \Phi^{EV}$ such that $fn(\Phi^{IV}) = fn(\Phi^{UV}) = fn(\Phi^{EV}) = rv(\Phi^{UV}) = rv(\Phi^{EV}) = \emptyset$, $rv(\Phi^{IV}) \subseteq rv(\mathsf{R}(V))$, and for all names $\tilde{s} = (s_1, \ldots, s_n)$ we have:*

1. *The tests Φ^{IV} and Φ^{UV} satisfy each of the conditions of Definition 4;*
2. *The additional conditions 5, 6, 7 and 8 below hold.*

Let $\tilde{r} = rv(\Phi^{IV})$, $\Phi_i^{IV} = \Phi^{IV}\{^{s_i}/_{v}, ^{\tilde{r}_i}/_{\tilde{r}}, ^{y_i}/_{y}\}$ and $X = fv(\Phi^{EV})\backslash dom(\mathsf{VP}_{\mathsf{n}}^{+}(s_1, \ldots, s_n))$

Soundness. *For all contexts C and processes B such that $C[\mathsf{VP}^+_{\mathsf{n}}(s_1, \ldots, s_n)] \Longrightarrow B$ and $\phi(B) \equiv \nu\tilde{n}.\sigma$, we have:*

$$\Phi^{EV}\sigma \wedge \Phi^{EV}\{^{x'}/_{x} \mid x \in X \backslash \tilde{y}\}\sigma \Rightarrow \tilde{w}\sigma \simeq \tilde{w}'\sigma \tag{5}$$

$$\bigwedge_{1 \leq i \leq n} \Phi_i^{IV}\sigma \wedge \Phi^{EV}\{^{\tilde{w}'}/_{\tilde{w}}\}\sigma \quad \Rightarrow \quad \tilde{w}\sigma \simeq \tilde{w}'\sigma \tag{6}$$

$$\Phi^{EV}\sigma \wedge \Phi^{EV}\{^{x'}/_{x} \mid x \in X \backslash \tilde{w}\}\sigma \Rightarrow \tilde{y}\sigma \simeq \tilde{y}'\sigma \tag{7}$$

Effectiveness. *There exists a context C and a process B such that $C[\mathsf{VP}^+_{\mathsf{n}}(s_1, \ldots, s_n)] \Longrightarrow B$, $\phi(B) \equiv \nu\tilde{n}.\sigma$ and*

$$\bigwedge_{1 \leq i \leq n} \Phi_i^{IV}\sigma \wedge \Phi^{UV}\sigma \wedge \Phi^{EV}\sigma \tag{8}$$

The test Φ^{EV} is instantiated by an observer with the bulletin board. Condition (5) ensures that, given a set of ballots $\tilde{y}\sigma$, provided by the environment, Φ^{EV} succeeds only for one list of voter public credentials. Condition (6) ensures that if a bulletin board contains the ballots of voters with public credentials $\tilde{w}\sigma$ then Φ^{EV} only holds on a permutation of these credentials. Condition (7) ensures that, given a set of credentials $\tilde{w}$, only one set of bulletin board entries $\tilde{y}$ are accepted by Φ^{EV} (observe that for such a strong requirement to hold we expect the voting specification's frame to contain a public key, to root trust). Finally, the effectiveness condition is similar to Condition (4) of the previous section.

Example 5. *The raising hands protocol satisfies eligibility verifiability. Let*

$$\Phi^{EV} \mathrel{\hat{=}} \bigwedge_{1 \leq i \leq n} \big(\mathsf{checksign}(x_{pkA}, w_i) =_E \mathsf{true} \wedge \pi_1(y_i) =_E \mathsf{getmsg}(w_i) \\ \wedge \mathsf{checksign}(\pi_1(y_1), \pi_2(y_i)) =_E \mathsf{true}\big)$$

We also need to slightly strengthen Φ^{IV} which is defined as

$$\Phi^{IV} \mathrel{\hat{=}} y =_E (\!|\mathsf{pk}(r_{skv}), \mathsf{sign}(r_{skv}, v)|\!) \wedge \mathsf{getmsg}(w) =_E \mathsf{pk}(r_{skv}) \\ \wedge \mathsf{checksign}(x_{pkA}, w) =_E \mathsf{true}$$

Conditions (1) – (4) can be proved as previously (Example 3). It remains to show that Conditions (5) – (8) hold.

(5) Suppose $\Phi^{EV}\sigma$ and $\Phi^{EV}\{^{x'}/_{x} \mid x \in X \backslash \tilde{y}\}\sigma$ hold; for all $1 \leq i \leq n$ we have

$$\mathsf{getmsg}(w_i)\sigma =_E \pi_1(y_i)\sigma \wedge \pi_1(y_i)\sigma =_E \mathsf{getmsg}(w_i')\sigma \\ \wedge \mathsf{checksign}(w_i, x_{pkA})\sigma =_E \mathsf{true} \wedge \mathsf{checksign}(w_i', x_{pkA})\sigma =_E \mathsf{true}$$

By inspection of the equational theory we have that $\tilde{w}_i\sigma =_E \tilde{w}_i'\sigma$.

(6) *Suppose that* $\bigwedge_{1\leq i\leq n}\Phi_i^{IV}\sigma$ *and* $\Phi^{EV}\sigma$ *hold; hence for all* $1 \leq i \leq n$ *we have*

$$\begin{aligned}&\mathsf{getmsg}(w_i)\sigma =_E \mathsf{getmsg}(w_i')\sigma\\ &\qquad\qquad \wedge \mathsf{checksign}(x_{pkA}, w_i)\sigma =_E \mathsf{checksign}(w_i', x_{pkA})\sigma =_E \mathsf{true}\end{aligned}$$

Again we have $\tilde{w}_i\sigma =_E \tilde{w}_i'\sigma$.

(7) *Suppose* $\Phi^{EV}\sigma \wedge \Phi^{EV}\{^{x'}/_x \mid x \in X\backslash\tilde{w}\}\sigma$ *hold; for all* $1 \leq i \leq n$ *we have*

$$\begin{aligned}&\pi_1(y_i)\sigma =_E \mathsf{getmsg}(w_i)\sigma =_E \pi_1(y_i')\sigma \wedge \mathsf{checksign}(x_{pkA}, w_i)\sigma =_E \mathsf{true} \wedge\\ &\mathsf{checksign}(\pi_1(y_i), \pi_2(y_i))\sigma =_E \mathsf{true} \wedge \mathsf{checksign}(\pi_1(y_i'), \pi_2(y_i'))\sigma =_E \mathsf{true}\end{aligned}$$

For any σ *such that* $C[\mathsf{VP}_{\mathsf{n}}^{+}(s_1, \ldots, s_n)] \Longrightarrow B$ *and* $\phi(B) \equiv \nu\tilde{n}.\sigma$ *we have that if* $\mathsf{checksign}(x_{pkA}, w_i)\sigma =_E \mathsf{true}$ *then* $\mathsf{getmsg}(w_i)\sigma =_E \mathsf{pk}(skv_j)$ *for some* $j \in [1..n]$. *Moreover, if* $\mathsf{checksign}(\mathsf{pk}(skv_j), \pi_2(x))\sigma =_E \mathsf{true}$ *then* $\mathsf{getmsg}(\pi_2(y_i))\sigma =_E s_j$. *Hence we have that for all* $1 \leq i \leq n$ *that* $\pi_2(y_i)\sigma =_E \pi_2(y_i')\sigma$. *Finally we conclude* $\tilde{y}\sigma =_E \tilde{y}'\sigma$.

Case studies: FOO and Helios 2.0. Neither FOO nor Helios use public voting credentials in a manner suitable for eligibility verifiability. In FOO, the administrator is responsible for ensuring eligibility, that is, checking the validity of the voter's ballots; whereas in Helios, there are no public voting credentials. It follows immediately that Condition (7), in particular, cannot be satisfied.

5.1 Case study: JCJ-Civitas

The protocol due to Juels, Catalano & Jakobsson [18] is based on mixnets and has been implemented by Clarkson, Chong & Myers [13, 12] as an open-source voting system called Civitas. The schemes, which we call JCJ-Civitas, are the first to provide election verifiability.

How JCJ-Civitas works. An election is created by naming a set of registrars and talliers. The protocol is divided into four phases: *setup*, *registration*, *voting* and *tallying*. We now detail the steps of the protocol, starting with the setup phase.

1. The registrars (respectively talliers) run a protocol which constructs a public key pair and distributes a share of the secret part amongst the registrars' (respectively talliers'). The public part $\mathsf{pk}(sk_T)$ (respectively $\mathsf{pk}(sk_R)$) of the key is then published. In addition, the registrars construct a distributed signing key pair ssk_R, $\mathsf{pk}(ssk_R)$.

The registration phase then proceeds as follows.

2. The registrars generate and distribute voter credentials: a private part d and a public part $\mathsf{penc}(\mathsf{pk}(sk_R), m'', d)$ (the probabilistic encryption of d under the registrars' public key $\mathsf{pk}(sk_R)$). This is done in a distributed manner, so that no registrar learns the value of any private credential d.

3. The registrars publish the signed public voter credentials.

4. The registrars announce the candidate list $\tilde{t} = (t_1, \ldots, t_l)$.

The protocol then enters the voting phase.

5. Each voter selects her vote $s \in \tilde{t}$ and computes two ciphertexts $M = \mathsf{penc}(\mathsf{pk}(sk_T), m, s)$ and $M' = \mathsf{penc}(\mathsf{pk}(sk_R), m', d)$ where m, m' are nonces. M contains her vote and M' her credential. In addition, the voter constructs a non-interactive zero-knowledge proof of knowledge demonstrating the correct construction of her ciphertexts and validity of the candidate ($s \in \tilde{t}$). (The ZKP provides protection against coercion resistance, by preventing forced abstention attacks via a *write in*, and binds the two ciphertexts for eligibility verifiability.) The voter derives her ballot as the triple consisting of her ciphertexts and zero-knowledge proof and posts it to the bulletin board.

After some predefined deadline the tallying phase commences in order to compute the election outcome.

6. The talliers read the n' ballots posted to the bulletin board by voters (that is, the triples consisting of the two ciphertexts and the zero-knowledge proof) and discards any entries for which the zero-knowledge proof does not hold.

7. The elimination of re-votes is performed on the ballots using pairwise *plaintext equality tests* (PET) on the ciphertexts containing private voter credentials. (A PET [16] is a cryptographic predicate which allows a keyholder to provide a proof that two ciphertexts contain the same plaintext.) Re-vote elimination is performed in a verifiable manner with respect to some publicly defined policy, *e.g.*, by the order of ballots on the bulletin board.

8. The talliers perform a verifiable re-encryption mix on the ballots (ballots consist of a vote ciphertext and a public credential ciphertext; the link between both is preserved by the mix.) The mix ensures that a voter cannot trace her vote, allowing the protocol to achieve coercion-resistance.

9. The talliers perform a verifiable re-encryption mix on the list of public credentials published by the registrar. This mix anonymises public voter credentials, breaking any link with the voter for privacy purposes.

10. Ballots based on invalid credentials are weeded using PETs between the mixed ballots and the mixed public credentials. Both have been posted to the bulletin board. (Using PETs the correctness of weeding is verifiable.)

11. Finally, the talliers perform a verifiable decryption and publish the result.

Equational theory. The protocol uses a variant of the ElGamal encryption scheme [18]. Accordingly we adopt the signature and associated equational theory from the Helios case study. The zero-knowledge proof demonstrating correct construction of the voter's ciphertexts is modelled by the equation

$$\mathsf{checkBallot}(\mathsf{ballotPf}(x_{\mathsf{pk}}, x_{\mathsf{rand}}, x_{\mathsf{text}}, x'_{\mathsf{pk}}, x'_{\mathsf{rand}}, x'_{\mathsf{text}}), \\ \mathsf{penc}(x_{\mathsf{pk}}, x_{\mathsf{rand}}, x_{\mathsf{text}}), \mathsf{penc}(x'_{\mathsf{pk}}, x'_{\mathsf{rand}}, x'_{\mathsf{text}})) = \mathsf{true}$$

(For simplicity the zero-knowledge proof does not demonstrate that the voter's vote s is a valid vote, that is, $s \in \tilde{t}$; this is of importance for privacy properties, not verifiability.) Plaintext equivalence tests are modelled by the equation

$$\mathsf{pet}(\mathsf{petPf}(x_{\mathsf{sk}}, ciph, ciph'), ciph, ciph') = \mathsf{true}$$

where $ciph \mathrel{\hat{=}} \mathsf{penc}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{rand}}, x_{\mathsf{text}})$ and $ciph' \mathrel{\hat{=}} \mathsf{penc}(\mathsf{pk}(x_{\mathsf{sk}}), x'_{\mathsf{rand}}, x_{\mathsf{text}})$. Re-encryption is defined with respect to the standard equation

$$\mathsf{renc}(y_{\mathsf{rand}}, \mathsf{penc}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{rand}}, x_{\mathsf{text}})) = \mathsf{penc}(\mathsf{pk}(x_{\mathsf{sk}}), f(x_{\mathsf{rand}}, y_{\mathsf{rand}}), x_{\mathsf{text}}).$$

In addition we consider verifiable re-encryption mixnets and introduce for each permutation χ on $\{1, \ldots, n\}$ the equation:

$$\mathsf{checkMix}(\mathsf{mixPf}(x_{\mathsf{ciph},1}, \ldots, x_{\mathsf{ciph},n}, \\ ciph_1, \ldots, ciph_n, z_{\mathsf{rand},1}, \ldots, z_{\mathsf{rand},n}), \\ x_{\mathsf{ciph},1}, \ldots, x_{\mathsf{ciph},n}, ciph_1, \ldots, ciph_n) = \mathsf{true}$$

where $ciph_i \mathrel{\hat{=}} \mathsf{renc}(z_{\mathsf{rand},i}, x_{\mathsf{ciph},\chi(i)})$. We also define re-encryption with respect to pairs of ciphertexts and introduce for each permutation χ on $\{1, \ldots, n\}$ the equation:

$$\mathsf{checkMixPair}(\mathsf{mixPairPf}(\langle x_1, x'_1\rangle, \ldots, \langle x_n, x'_n\rangle, \\ \langle c_1, c'_1\rangle, \ldots, \langle c_n, c'_n\rangle, \langle z_1, z'_1\rangle, \ldots, \langle z_n, z'_n\rangle), \\ \langle x_1, x'_1\rangle, \ldots, \langle x_n, x'_n\rangle, \langle c_1, c'_1\rangle, \ldots, \langle c_n, c'_n\rangle) = \mathsf{true}$$

where $c_i \mathrel{\hat{=}} \mathsf{renc}(z_i, x_{\chi(i)})$ and $c'_i \mathrel{\hat{=}} \mathsf{renc}(z'_i, x'_{\chi(i)})$.

The following lemmata demonstrate useful properties of our equational theory. We make use of the notation $\tilde{M} \mathrel{\dot{\simeq}} \tilde{M}'$ to denote that the ciphertext tuples $\tilde{M}$, $\tilde{M}'$ are defined over the same plaintexts with respect to some public key K, that is, we have $\tilde{M} =_E (\mathsf{penc}(K, R_1, N_1), \ldots \mathsf{penc}(K, R_n, N_n))$, $\tilde{M}' =_E (\mathsf{penc}(K, R'_1, N'_1), \ldots \mathsf{penc}(K, R'_n, N'_n))$ for some tuples $\tilde{N}, \tilde{N}', \tilde{R}, \tilde{R}'$ and there exists a permutation χ defined over $\{1, \ldots, n\}$ such that for all $1 \leq i \leq n$ we have $N_i =_E N'_{\chi(i)}$. The relation $\mathrel{\dot{\simeq}}$ is trivially seen to be an equivalence relation. Moreover, if $\tilde{M} \mathrel{\dot{\simeq}} \tilde{N}$ and $\tilde{M} \simeq \tilde{M}'$, then $M' \mathrel{\dot{\simeq}} \tilde{N}$.

Lemma 1. *Given terms* L, M, N*, if* $\mathsf{pet}(L, M, N) =_E \mathsf{true}$*, then* $M \overset{\bullet}{\simeq} N$*.*

Lemma 2. *Given terms* $L, \tilde{M}, \tilde{N}$*, if* $\mathsf{checkMix}(L, \tilde{M}, \tilde{N}) =_E \mathsf{true}$*, then* $\tilde{M} \overset{\bullet}{\simeq} \tilde{N}$*.*

Lemma 3. *Given terms* $L, \tilde{M}, \tilde{N}$*, if* $\mathsf{checkMixPair}(L, \tilde{M}, \tilde{N}) =_E \mathsf{true}$*, then* $(\pi_i(M_1), \ldots, \pi_i(M_{|\tilde{M}|})) \overset{\bullet}{\simeq} (\pi_i(N_1), \ldots, \pi_i(N_{|\tilde{N}|}))$*.*

Model in applied pi. We make the following trust assumptions for verifiability:

- The voter is able to construct her ballot; that is, she is able to generate nonces m, m', construct a pair of ciphertexts and generate a zero-knowledge proof.
- The registrar constructs distinct credentials d for each voter and constructs the voter's public credential correctly. (The latter assumption can be dropped if the registrar provides a proof that the public credential is correctly formed [18].) The registrar also keeps the private part of the signing key secret.

Although neither voters nor observers can verify that the registrars adhere to such expectations, they trust them because trust is distributed. The trusted components are modelled by the voting process specification $\langle A_{\mathsf{jcj}}, V_{\mathsf{jcj}} \rangle$ (Definition **??**). The context A_{jcj} publishes public keys and defines a sub-process R to model the registrar. The registrar R constructs a fresh private credential d and sends the private credential along with the signed public part (that is, $\mathsf{sign}(ssk_R, \mathsf{penc}(x_{pk_R}, m'', d))$) to the voter; the registrar also publishes the signed public credential on the bulletin board. The voter V_{jcj} receives the private and public credentials from the registrar and constructs her ballot; that is, the pair of ciphertexts and a zero-knowledge proof demonstrating their correct construction.

Definition 8. *The voting process specification* $A_{\mathsf{jcj}}, V_{\mathsf{jcj}}$ *is defined where:*

$$
\begin{array}{rcl}
A_{\mathsf{jcj}} & \hat{=} & \nu\, a, ssk_R.(!R \mid \{{}^{\mathsf{pk}(sk_R)}/_{x_{\mathsf{pk_R}}}, {}^{\mathsf{pk}(ssk_R)}/_{x_{spk_R}}, {}^{\mathsf{pk}(sk_T)}/_{x_{\mathsf{pk_T}}}\} \mid \_) \\
V_{\mathsf{jcj}} & \hat{=} & \nu\, m, m'.a(x_{cred}). \\
 & & \mathit{let}\ ciph = \mathsf{penc}(x_{pk_T}, m, v)\ \mathit{in} \\
 & & \mathit{let}\ ciph' = \mathsf{penc}(x_{pk_R}, m', \pi_1(x_{cred}))\ \mathit{in} \\
 & & \mathit{let}\ zkp = \mathsf{ballotPf}(x_{pk_T}, m, v, x_{pk_R}, m', \pi_1(x_{cred}))\ \mathit{in} \\
 & & \overline{c}\langle (\!| ciph, ciph', zkp |\!) \rangle \\
R & \hat{=} & \nu\, d, m''.\, \mathit{let}\ sig = \mathsf{sign}(ssk_R, \mathsf{penc}(x_{pk_R}, m'', d))\ \mathit{in}\ \overline{a}\langle (\!| d, sig |\!) \rangle.\overline{c}\langle sig \rangle
\end{array}
$$

At the end of the election the bulletin board is represented by the frame. In our formalism we expect the frame to contain the substitution σ which defines the voters' public credentials as $w_1, \ldots, w_n$, public keys of the registrars as x_{pk_R}, x_{spk_R} and talliers' public key as x_{pk_T}. Triples $y_1, \ldots, y_n$ consisting of each voter's ciphertexts and zero-knowledge proofs. The mixed re-encryptions of the voter's ciphertexts $z_{\mathsf{bal},1}, \ldots, z_{\mathsf{bal},n}$ along with a proof $z_{\mathsf{mixPairPf}}$ that the

mix was performed correct. For verifiable decryption we assume $z_{\mathsf{decKey},i}$ is defined as a decryption key associated with the proof $z_{\mathsf{decPf},i}$. For the purposes of eligibility verifiability we also expect the mixed re-encryptions of the voter's public credentials $z_{\mathsf{cred},1},\ldots,z_{\mathsf{cred},1}$ along with a proof of correctness z_{mixPf}. For convenience a reordering $\hat{z}_{\mathsf{cred},1},\ldots,\hat{z}_{\mathsf{cred},n}$ of these re-encryptions is also computed. Finally, we expect PET proofs $z_{\mathsf{petPf},1},\ldots,z_{\mathsf{petPf},n}$ for the reencryption of the ciphertext constructed by the voter on her private credential (that is, the output of the verifiable mix in Step 8 of the protocol) and the reencryption of the voter's public credential constructed by the registrars (that is, the output of the mix in Step 9); such that the PET holds, that is, the pair of ciphertexts contain the same private credential. Accordingly we expect σ to be such that for all $1 \leq i \leq n$:

$$
\begin{array}{lcl}
w_i\sigma & = & \mathsf{sign}(ssk_R, c_i'') \\
x_{pk_R}\sigma & = & \mathsf{pk}(sk_R) \\
x_{spk_R}\sigma & = & \mathsf{pk}(ssk_R) \\
x_{pk_T}\sigma & = & \mathsf{pk}(sk_T) \\
y_i\sigma & = & (\!|c_i, c_i', \mathsf{ballotPf}(\mathsf{pk}(sk_T), m_i, s_i, \mathsf{pk}(sk_R), m_i', d_i)|\!) \\
z_{\mathsf{bal},i}\sigma & = & (\!|\mathsf{renc}(\hat{m}_i, c_{\chi(i)}), \mathsf{renc}(\hat{m}_i', c'_{\chi(i)})|\!) \\
z_{\mathsf{mixPairPf}}\sigma & = & \mathsf{pfMixPair}((\!|c_1, c_1'|\!), \ldots, (\!|c_n, c_n'|\!), (\!|\mathsf{renc}(\hat{m}_1, c_{\chi(1)}), \mathsf{renc}(\hat{m}_1', c'_{\chi(1)})|\!), \\
 & & \ldots, (\!|\mathsf{renc}(\hat{m}_n, c_{\chi(n)}), \mathsf{renc}(\hat{m}_n', c'_{\chi(n)})|\!), (\!|\hat{m}_1, \hat{m}_1'|\!), \ldots, (\!|\hat{m}_n, \hat{m}_n'|\!)) \\
z_{\mathsf{decKey},i}\sigma & = & \mathsf{decKey}(sk_T, \mathsf{renc}(\hat{m}_i, c_{\chi(i)})) \\
z_{\mathsf{decPf},i}\sigma & = & \mathsf{decKeyPf}(sk_T, \mathsf{renc}(\hat{m}_i, c_{\chi(i)}), \mathsf{decKey}(sk_T, \mathsf{renc}(\hat{m}_i, c_{\chi(i)}))) \\
z_{\mathsf{cred},i}\sigma & = & \mathsf{renc}(\hat{m}_i'', c''_{\chi'(i)}) \\
\hat{z}_{\mathsf{cred},i}\sigma & = & \mathsf{renc}(\hat{m}''_{\chi(\chi'^{-1}(i))}, c''_{\chi(i)}) \\
z_{\mathsf{mixPf}}\sigma & = & \mathsf{pfMix}(c_1'', \ldots, c_n'', \mathsf{renc}(\hat{m}_1'', c''_{\chi'(1)}), \ldots, \mathsf{renc}(\hat{m}_n'', c''_{\chi'(n)}), \hat{m}_1'', \ldots, \hat{m}_n'') \\
z_{\mathsf{petPf},i}\sigma & = & \mathsf{petPf}(sk_R, \mathsf{renc}(\hat{m}_i', c'_{\chi(i)}), \mathsf{renc}(\hat{m}''_{\chi(\chi'^{-1}(i))}, c''_{\chi(i)}))
\end{array}
$$

where $c_i \mathrel{\hat{=}} \mathsf{penc}(\mathsf{pk}(sk_T), m, s_i)$, $c_i' \mathrel{\hat{=}} \mathsf{penc}(\mathsf{pk}(sk_R), m', d_i)$, $c_i'' \mathrel{\hat{=}} \mathsf{penc}(\mathsf{pk}(sk_R), m'', d_i)$ and χ, χ' are permutations on $\{1,\ldots,n\}$.

Election verifiability. For the purpose of election verifiability we introduce the tests $\Phi^{IV}, \Phi^{UV}, \Phi^{EV}$. Without loss of generality suppose the recording function uses record variables $\tilde{r} = (r_{cred}, r_m, r_{m'}) = \mathrm{rv}(\mathsf{R}(V))$ (corresponding to the variable x_{cred} and names m, m' appearing in the process V). Accordingly, given $n \in \mathbb{N}$ we define:

$$
\begin{array}{lcl}
\Phi^{IV} & \mathrel{\hat{=}} & y =_E (\!|\mathsf{penc}(x_{\mathsf{pk_T}}, r_m, v), \mathsf{penc}(x_{\mathsf{pk_R}}, r_{m'}, \pi_1(r_{cred})), \\
 & & \quad \mathsf{ballotPf}(x_{\mathsf{pk_T}}, r_m, v, x_{\mathsf{pk_R}}, r_{m'}, \pi_1(r_{cred}))|\!) \wedge w = \pi_2(r_{cred}) \\
\Phi^{UV} & \mathrel{\hat{=}} & \mathsf{checkMixPair}(z_{\mathsf{mixPairPf}}, (\!|\pi_1(y_1), \pi_2(y_1)|\!), \ldots, (\!|\pi_1(y_n), \pi_2(y_n)|\!), \\
 & & \quad z_{\mathsf{bal},1}, \ldots, z_{\mathsf{bal},n}) =_E \mathsf{true} \\
 & & \wedge \bigwedge_{i=1}^{n} \mathsf{dec}(z_{\mathsf{decKey},i}, \pi_1(z_{\mathsf{bal},i})) =_E v_i \\
 & & \wedge \bigwedge_{i=1}^{n} \mathsf{checkDecKeyPf}(x_{\mathsf{pk_T}}, \pi_1(z_{\mathsf{bal},i}), z_{\mathsf{decKey},i}, z_{\mathsf{decPf},i}) =_E \mathsf{true} \\
\Phi^{EV} & \mathrel{\hat{=}} & \bigwedge_{i=1}^{n} \mathsf{checkBallot}(\pi_3(y_i), \pi_1(y_i), \pi_2(y_i)) \\
 & & \wedge \mathsf{checkMixPair}(z_{\mathsf{mixPairPf}}, (\!|\pi_1(y_1), \pi_2(y_1)|\!), \ldots, (\!|\pi_1(y_n), \pi_2(y_n)|\!), \\
 & & \quad z_{\mathsf{bal},1}, \ldots, z_{\mathsf{bal},n}) =_E \mathsf{true}
\end{array}
$$

$\wedge \bigwedge_{i=1}^{n} \mathsf{pet}(z_{\mathsf{petPf},i}, \pi_2(z_{\mathsf{bal},i}), \hat{z}_{\mathsf{cred},i}) =_E \mathsf{true}$
$\wedge \langle\!| z_{\mathsf{cred},1}, \ldots, z_{\mathsf{cred},n} |\!\rangle \simeq \langle\!| \hat{z}_{\mathsf{cred},1}, \ldots, \hat{z}_{\mathsf{cred},n} |\!\rangle$
$\wedge \mathsf{checkMix}(z_{\mathsf{mixPf}}, \mathsf{getmsg}(w_1), \ldots, \mathsf{getmsg}(w_n), z_{\mathsf{cred},1}, \ldots, z_{\mathsf{cred},n}) =_E \mathsf{true}$
$\wedge \bigwedge_{i=1}^{n} \mathsf{checksign}(x_{spk_R}, w_i)$

The test Φ^{IV} checks that the voter's ballot and public credential are recorded on the bulletin board. The test Φ^{UV} checks that the tally is correctly computed; that is, the mix is checked, the validity of decryption keys have been verified and the decrypted tally corresponds to the declared outcome. Finally, the test Φ^{EV} checks that only eligible ballots are considered; that is, ballots are correctly formed, mixes have been handled in suitable manner, PETs have been verified and only authentic public voter credentials are considered.

Theorem 3. *$\langle A_{\mathsf{jcj}}, V_{\mathsf{jcj}} \rangle$ satisfies election verifiability.*

Proof. Suppose $n \in \mathbb{N}$ and the tests $\Phi^{IV}, \Phi^{UV}, \Phi^{EV}$ are given above. We will now show that for all names $\tilde{s} = (s_1, \ldots, s_n)$ that the conditions of Definition 7 hold.

(1) Suppose C is a context, B is a process and i, j are integers such that $C[\mathsf{VP}^{+}_{\mathsf{n}}(s_1, \ldots, s_n)] \Longrightarrow B$, $\phi(B) \equiv \nu\tilde{n}.\sigma$ and $\Phi^{IV}\{^{s_i}/_{v}, ^{\tilde{r}_i}/_{\tilde{r}}\}\sigma \wedge \Phi^{IV}\{^{s_j}/_{v}, ^{\tilde{r}_j}/_{\tilde{r}}\}\sigma$. It follows that $\pi_1(y)\sigma =_E \mathsf{penc}(x_{\mathsf{pk_T}}, r_{m,i}, s_i)\sigma =_E \mathsf{penc}(x_{\mathsf{pk_T}}, r_{m,j}, s_j)\sigma$ and by inspection of the equational theory it is the case that $r_{m,i}\sigma = r_{m,j}\sigma$. Since the record variables $r_{m,i}$, $r_{m,j}$ are handles for fresh nonces created by name restriction in the voter process it follows immediately from $r_{m,i}\sigma = r_{m,j}\sigma$ that $i = j$.

(2) We prove a stronger result, namely for any σ the condition holds. Suppose $\Phi^{UV}\sigma \wedge \Phi^{UV}\{^{\tilde{v}'}/_{\tilde{v}}\}\sigma$ and hence

$$\bigwedge_{i=1}^{n} \mathsf{dec}(z_{\mathsf{decKey},i}, \pi_1(z_{\mathsf{bal},i}))\sigma =_E v_i\sigma =_E v_i'\sigma.$$

It follows immediately that $\tilde{v}\sigma =_E \tilde{v}'\sigma$.

(3) Again, we will show that the condition holds for all substitutions σ. Suppose $\Phi^{IV}\{^{s_i}/_{v}, ^{\tilde{r}_i}/_{\tilde{r}}, ^{y_i}/_{y}\}\sigma$ holds for $1 \leq i \leq n$ and hence

$$\bigwedge_{1 \leq i \leq n} \pi_1(y_i)\sigma =_E \mathsf{penc}(x_{\mathsf{pk_T}}, r_{m,i}, s_i)\sigma.$$

Moreover suppose $\Phi^{UV}\sigma$ holds and therefore

$$\mathsf{checkMixPair}(z_{\mathsf{mixPairPf}}, \langle\!|\pi_1(y_1), \pi_2(y_1)|\!\rangle, \ldots, \langle\!|\pi_1(y_n), \pi_2(y_n)|\!\rangle, z_{\mathsf{bal},1}, \ldots, z_{\mathsf{bal},n})\sigma =_E \mathsf{true}$$

holds. By inspection of the equational theory we have

$$\pi_1(z_{\mathsf{bal},i})\sigma =_E \mathsf{penc}(x_{\mathsf{pk_T}}, f(r_{m,\chi(i)}, R_i), s_{\chi(i)})\sigma$$

for some permutation χ defined over $\{1,\ldots,n\}$ and terms $R_1,\ldots,R_n$ (note $R_1,\ldots,R_n$ appear in $z_{\mathsf{mixPairPf}}\sigma$). By our hypothesis, we also have for all $1 \leq i \leq n$ that

$$\mathsf{checkDecKeyPf}(x_{\mathsf{pk_T}}, \pi_1(z_{\mathsf{bal},i}), z_{\mathsf{decKey},i}, z_{\mathsf{decPf},i})\sigma =_E \mathsf{true}$$

and hence $z_{\mathsf{decKey},i}\sigma$ is a decryption key for $\pi_1(z_{\mathsf{bal},i})\sigma$. It follows that

$$\bigwedge_{1\leq i\leq n} \mathsf{dec}(z_{\mathsf{decKey},i}, \pi_1(z_{\mathsf{bal},i}))\sigma =_E s_{\chi(i)}$$

Finally, by hypothesis, we also have

$$\bigwedge_{1\leq i\leq n} \mathsf{dec}(z_{\mathsf{decKey},i}, \pi_1(z_{\mathsf{bal},i}))\sigma =_E v_i\sigma$$

and hence it follows that $\tilde{s} \simeq \tilde{v}$.

(4) We prove a stronger result, namely Condition 8 below.

(5) Suppose C is a context and B is a process such that $C[\mathsf{VP}^+_{\mathsf{n}}(s_1,\ldots,s_n)] \implies B$, $\phi(B) \equiv \nu\tilde{n}.\sigma$, and $\Phi^{EV}\sigma \wedge \Phi^{EV}\{^{x'}/_x \mid x \in X\backslash\tilde{y}\}\sigma$. We have for all $1 \leq i \leq n$ that $\mathsf{checkBallot}(\pi_3(y_i), \pi_1(y_i), \pi_2(y_i))\sigma =_E \mathsf{true}$ and it follows by inspection of the equational theory that

$$\pi_2(y_i)\sigma =_E \mathsf{penc}(K_i, S_i, M_i)$$

for some terms K_i, S_i, M_i. Since $\mathsf{checkMixPair}(z_{\mathsf{mixPairPf}}, (\!|\pi_1(y_1), \pi_2(y_1)|\!), \ldots, (\!|\pi_1(y_n), \pi_2(y_n)|\!), z_{\mathsf{bal},1}, \ldots, z_{\mathsf{bal},n})\sigma =_E \mathsf{true}$ and $\mathsf{checkMixPair}(z_{\mathsf{mixPairPf}}', (\!|\pi_1(y_1), \pi_2(y_1)|\!), \ldots, (\!|\pi_1(y_n), \pi_2(y_n)|\!), z'_{\mathsf{bal},1}, \ldots, z'_{\mathsf{bal},n})\sigma =_E \mathsf{true}$, it follows by Lemma 3 and transitivity of $\overset{\bullet}{\simeq}$ that

$$(\pi_2(z_{\mathsf{bal},1}), \ldots, \pi_2(z_{\mathsf{bal},n}))\sigma \overset{\bullet}{\simeq} (\pi_2(z'_{\mathsf{bal},1}), \ldots, \pi_2(z'_{\mathsf{bal},n}))\sigma.$$

Moreover, we have for all $1 \leq i \leq n$ that $\mathsf{pet}(z_{\mathsf{petPf},i}, \pi_2(z_{\mathsf{bal},i}), \hat{z}_{\mathsf{cred},i})\sigma =_E \mathsf{true}$ and $\mathsf{pet}(z'_{\mathsf{petPf},i}, \pi_2(z'_{\mathsf{bal},i}), \hat{z}'_{\mathsf{cred},i})\sigma =_E \mathsf{true}$; by Lemma 1 it follows that

$$(\hat{z}_{\mathsf{cred},1}, \ldots, \hat{z}_{\mathsf{cred},n})\sigma \overset{\bullet}{\simeq} (\hat{z}'_{\mathsf{cred},1}, \ldots, \hat{z}'_{\mathsf{cred},n})\sigma.$$

We have $(\!|z_{\mathsf{cred},1}, \ldots, z_{\mathsf{cred},n}|\!)\sigma \simeq (\!|\hat{z}_{\mathsf{cred},1}, \ldots, \hat{z}_{\mathsf{cred},n}|\!)\sigma$, $(\!|z'_{\mathsf{cred},1}, \ldots, z'_{\mathsf{cred},n}|\!)\sigma \simeq (\!|\hat{z}'_{\mathsf{cred},1}, \ldots, \hat{z}'_{\mathsf{cred},n}|\!)\sigma$ and hence we trivially derive

$$(z_{\mathsf{cred},1}, \ldots, z_{\mathsf{cred},n})\sigma \overset{\bullet}{\simeq} (z'_{\mathsf{cred},1}, \ldots, z'_{\mathsf{cred},n})\sigma.$$

Since $\mathsf{checkMix}(z_{\mathsf{mixPf}}, \mathsf{getmsg}(w_1), \ldots, \mathsf{getmsg}(w_n), z_{\mathsf{cred},1}, \ldots, z_{\mathsf{cred},n})\sigma =_E \mathsf{true}$ and $\mathsf{checkMix}(z_{\mathsf{mixPf}}', \mathsf{getmsg}(w'_1), \ldots, \mathsf{getmsg}(w'_n), z'_{\mathsf{cred},1}, \ldots, z'_{\mathsf{cred},n})\sigma =_E \mathsf{true}$; it follows by Lemma 2 that

$$(\mathsf{getmsg}(w_1), \ldots, \mathsf{getmsg}(w_n))\sigma \overset{\bullet}{\simeq} (\mathsf{getmsg}(w'_1), \ldots, \mathsf{getmsg}(w'_n))\sigma.$$

We have for all $1 \leq i \leq n$ that $\mathsf{checksign}(x_{spk_R}, w_i)\sigma =_E \mathsf{true}$ and $\mathsf{checksign}(x_{spk_R}, w'_i)\sigma =_E \mathsf{true}$ where $x_{spk_R}\sigma = \mathsf{pk}(ssk_R)$ and $ssk_R \in \tilde{n}$. By inspection of the equational theory it is the case that $w_i\sigma =_E \mathsf{sign}(ssk_R, M_i)\sigma$ and $w'_i\sigma =_E \mathsf{sign}(ssk_R, M'_i)\sigma$ for some terms M_i, M'_i. Furthermore, since for all $1 \leq i \leq n$ we have $\mathsf{getmsg}(w_i)\sigma =_E M_i$, $\mathsf{getmsg}(w'_i)\sigma =_E M'_i$ and because $(\mathsf{getmsg}(w_1), \ldots, \mathsf{getmsg}(w_n))\sigma \stackrel{\bullet}{\simeq} (\mathsf{getmsg}(w'_1), \ldots, \mathsf{getmsg}(w'_n))\sigma$, it follows that $\tilde{M} \stackrel{\bullet}{\simeq} \tilde{M}'$. Now, since the signing key is under restriction, and by inspection of the voting process and its possible outputs, it follows that for all $1 \leq i \leq n$ we have

$$\mathsf{getmsg}(w_i)\sigma =_E \mathsf{penc}(\mathsf{pk}(sk_R), m''_{\chi(i)}, d_{\chi(i)})$$
$$\mathsf{getmsg}(w'_i)\sigma =_E \mathsf{penc}(\mathsf{pk}(sk_R), m''_{\chi'(i)}, d_{\chi'(i)})$$

where d_i, m''_i are names under restriction in the registrar process R, $x_{pk_R}\sigma =_E \mathsf{pk}(sk_R)$ and χ, χ' are permutations defined over $\{1, \ldots, n\}$. Finally we conclude $\tilde{w}\sigma \simeq \tilde{w}'\sigma$.

(6) Suppose C is a context and B is a process such that $C[\mathsf{VP}^+_{\mathsf{n}}(s_1, \ldots, s_n)] \Longrightarrow B$, $\phi(B) \equiv \nu\tilde{n}.\sigma$, and $\bigwedge_{1 \leq i \leq n} \Phi^{IV}_i\sigma \wedge \Phi^{EV}\{^{\tilde{w}'}/_{\tilde{w}}\}\sigma$ holds. We have for all $1 \leq i \leq n$ that $w_i\sigma = \pi_2(r_{cred_i})\sigma$ and by inspection of the voting process we have

$$\tilde{w}\sigma =_E (\mathsf{sign}(ssk_R, \mathsf{penc}(\mathsf{pk}(sk_R), m''_1, d_1)), \ldots, \mathsf{sign}(ssk_R, \mathsf{penc}(\mathsf{pk}(sk_R), m''_n, d_n))).$$

In addition we have $\pi_2(y_i)\sigma =_E \mathsf{penc}(\mathsf{pk}(sk_R), m'_i, d_i)$ for all $1 \leq i \leq n$ and by similar reasoning to the above (see Condition 5.1) we derive $\tilde{w}\sigma \simeq \tilde{w}'\sigma$.

(7) Suppose C is a context and B is a process such that $C[\mathsf{VP}^+_{\mathsf{n}}(s_1, \ldots, s_n)] \Longrightarrow B$, $\phi(B) \equiv \nu\tilde{n}.\sigma$, and $\Phi^{EV}\sigma \wedge \Phi^{EV}\{^{x'}/_x \mid x \in X \backslash \tilde{w}\}\sigma$. We have for all $1 \leq i \leq n$ that $\mathsf{checksign}(x_{spk_R}, w_i)\sigma =_E \mathsf{true}$ where $x_{spk_R}\sigma = \mathsf{pk}(ssk_R)$ and $ssk_R \in \tilde{n}$. By inspection of the equational theory it is the case that

$$w_i\sigma =_E \mathsf{sign}(ssk_R, M_i)\sigma$$

for some term M_i. Since the signing key is under restriction, and by inspection of the voting process, it follows that for all $1 \leq i \leq n$ we have

$$M_i =_E \mathsf{penc}(\mathsf{pk}(sk_R), m''_i, d_i)$$

where d_i, m''_i are names under restriction in the registrar process R and $x_{pk_R}\sigma =_E \mathsf{pk}(sk_R)$. Since $\mathsf{checkMix}(z_{\mathsf{mixPf}}, \mathsf{getmsg}(w_1), \ldots, \mathsf{getmsg}(w_n), z_{\mathsf{cred},1}, \ldots, z_{\mathsf{cred},n})\sigma =_E \mathsf{true}$, $\mathsf{checkMix}(z_{\mathsf{mixPf}}{}', \mathsf{getmsg}(w_1), \ldots, \mathsf{getmsg}(w_n), z'_{\mathsf{cred},1}, \ldots, z'_{\mathsf{cred},n})\sigma =_E \mathsf{true}$ and for all $1 \leq i \leq n$ we have $\mathsf{getmsg}(w_i)\sigma =_E M_i$ it follows that

$$(z_{\mathsf{cred},1}, \ldots, z_{\mathsf{cred},n})\sigma \stackrel{\bullet}{\simeq} (z'_{\mathsf{cred},1}, \ldots, z'_{\mathsf{cred},n})\sigma$$

by Lemma 2. We have $(\!|z_{\mathsf{cred},1},\ldots,z_{\mathsf{cred},n}|\!)\sigma \simeq (\!|\hat{z}_{\mathsf{cred},1},\ldots,\hat{z}_{\mathsf{cred},n}|\!)\sigma$ and $(\!|z'_{\mathsf{cred},1},\ldots,z'_{\mathsf{cred},n}|\!)\sigma \simeq (\!|\hat{z}'_{\mathsf{cred},1},\ldots,\hat{z}'_{\mathsf{cred},n}|\!)\sigma$; it trivially follows that

$$(\hat{z}_{\mathsf{cred},1},\ldots,\hat{z}_{\mathsf{cred},n})\sigma \stackrel{\bullet}{\simeq} (\hat{z}'_{\mathsf{cred},1},\ldots,\hat{z}'_{\mathsf{cred},n})\sigma.$$

Moreover, we have for all $1 \leq i \leq n$ that $\mathsf{pet}(z_{\mathsf{petPf},i},\pi_2(z_{\mathsf{bal},i}),\hat{z}_{\mathsf{cred},i})\sigma =_E \mathsf{true}$ and $\mathsf{pet}(z'_{\mathsf{petPf},i},\pi_2(z'_{\mathsf{bal},i}),\hat{z}'_{\mathsf{cred},i})\sigma =_E \mathsf{true}$; hence by Lemma 1 it follows that

$$(\pi_2(z_{\mathsf{bal},1}),\ldots,\pi_2(z_{\mathsf{bal},n}))\sigma \stackrel{\bullet}{\simeq} (\pi_2(z'_{\mathsf{bal},1}),\ldots,\pi_2(z'_{\mathsf{bal},n}))\sigma.$$

By $\mathsf{checkMixPair}(z_{\mathsf{mixPairPf}},(\!|\pi_1(y_1),\pi_2(y_1)|\!),\ldots,(\!|\pi_1(y_n),\pi_2(y_n)|\!),z_{\mathsf{bal},1},\ldots,z_{\mathsf{bal},n})\sigma =_E \mathsf{true}$, $\mathsf{checkMixPair}(z_{\mathsf{mixPairPf}}{}',(\!|\pi_1(y'_1),\pi_2(y'_1)|\!),\ldots,(\!|\pi_1(y'_n),\pi_2(y'_n)|\!),z'_{\mathsf{bal},1},\ldots,z'_{\mathsf{bal},n})\sigma =_E \mathsf{true}$ and Lemma 3 we have

$$(\pi_2(y_1),\ldots,\pi_2(y_n))\sigma \stackrel{\bullet}{\simeq} (\pi_2(y'_1),\ldots,\pi_2(y'_n))\sigma.$$

We have for all $1 \leq i \leq n$ that $\mathsf{checkBallot}(\pi_3(y_i),\pi_1(y_i),\pi_2(y_i))\sigma =_E \mathsf{true}$ and $\mathsf{checkBallot}(\pi_3(y'_i),\pi_1(y'_i),\pi_2(y'_i))\sigma =_E \mathsf{true}$. By inspection of the equational theory and because $(\pi_2(y_1),\ldots,\pi_2(y_n))\sigma \stackrel{\bullet}{\simeq} (\pi_2(y'_1),\ldots,\pi_2(y'_n))\sigma \stackrel{\bullet}{\simeq} (\mathsf{penc}(\mathsf{pk}(sk_R),m''_1,d_1),\ldots,\mathsf{penc}(\mathsf{pk}(sk_R),m''_n,d_n))$ it is the case that

$$\begin{array}{lcl}\pi_3(y_i)\sigma & =_E & \mathsf{ballotPf}(PK_{T_i},R_i,N_i,\mathsf{pk}(sk_R),S_i,d_{\chi(i)})\\ \pi_3(y'_i)\sigma & =_E & \mathsf{ballotPf}(PK'_{T_i},R'_i,N'_i,\mathsf{pk}(sk_R),S'_i,d_{\chi'(i)})\end{array}$$

for some terms $PK_{T_i},R_i,N_i,S_i,PK'_{T_i},R'_i,N'_i,S'_i$ and permutations χ,χ' defined over $\{1,\ldots,n\}$. Since for all $1 \leq i \leq n$ the name d_i is under restriction in the voting process specification, it follows that

$$\begin{array}{lcl}\pi_3(y_i)\sigma & =_E & \mathsf{ballotPf}(\mathsf{pk}(sk_T),m_{\chi(i)},s_{\chi(i)},\mathsf{pk}(sk_R),m'_{\chi(i)},d_{\chi(i)})\\ \pi_3(y'_i)\sigma & =_E & \mathsf{ballotPf}(\mathsf{pk}(sk_T),m_{\chi'(i)},s_{\chi'(i)},\mathsf{pk}(sk_R),m'_{\chi'(i)},d_{\chi'(i)})\end{array}$$

(that is, $\pi_3(y_i)\sigma$, $\pi_3(y'_i)\sigma$ are the zero-knowledge proofs output by the voters) and moreover by the validity of the proof, we have

$$\begin{array}{lcl}\pi_1(y_i)\sigma & =_E & \mathsf{penc}(\mathsf{pk}(sk_T),m_{\chi(i)},s_{\chi(i)})\\ \pi_1(y'_i)\sigma & =_E & \mathsf{penc}(\mathsf{pk}(sk_T),m_{\chi'(i)},s_{\chi'(i)})\end{array}$$

$$\begin{array}{lcl}\pi_2(y_i)\sigma & =_E & \mathsf{penc}(\mathsf{pk}(sk_R),m'_{\chi(i)},d_{\chi(i)})\\ \pi_2(y'_i)\sigma & =_E & \mathsf{penc}(\mathsf{pk}(sk_R),m'_{\chi'(i)},d_{\chi'(i)})\end{array}$$

Finally we conclude $\tilde{y}\sigma \simeq \tilde{y}'\sigma$. (Formally we should also show that $|\tilde{y}| = |\tilde{y}'|$. We omitted this detail from our test Φ^{EV} for simplicity, however, in this instance it could be incorporated with the additional conjunct $y = (\!|\pi_1(y),\pi_2(y),\pi_3(y)|\!)$.)

(8) This can be witnessed by modelling the complete JCJ-Civitas protocol as the context $C[\_]$. □

6 Conclusion

We present a symbolic definition of election verifiability which allows us to precisely identify which parts of a voting system need to be trusted for verifiability. The suitability of systems can then be evaluated and compared on the basis of trust assumptions. We also consider eligibility verifiability, an aspect of verifiability that is often neglected and satisfied by only a few protocols, but nonetheless an essential mechanism to detect ballot stuffing. We have applied our definition to three protocols: FOO, which uses blind signatures; Helios 2.0, which is based on homomorphic encryption, and JCJ-Civitas, which uses mixnets and anonymous credentials. For each of these protocols we discuss the trust assumptions that a voter or an observer needs to make for the protocol to be verifiable. Since Helios 2.0 and JCJ-Civitas have been implemented and deployed, we believe our formalisation is suitable for analysing real world election systems.

Acknowledgements

We are particularly grateful to Michael Clarkson for careful reading of an earlier draft, and for his perceptive questions and comments.

References

[1] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *POPL'01: Proceedings of the 28th ACM Symposium on Principles of Programming Languages*, pages 104–115, New York, USA, 2001. ACM.

[2] B. Adida. *Advances in Cryptographic Voting Systems*. PhD thesis, MIT, 2006.

[3] B. Adida. Helios: Web-based open-audit voting. In *Proceedings of the Seventeenth Usenix Security Symposium*, pages 335–348. USENIX Association, 2008.

[4] B. Adida, O. de Marneffe, O. Pereira, and J.-J. Quisquater. Electing a university president using open-audit voting: Analysis of real-world use of Helios. In *Electronic Voting Technology/Workshop on Trustworthy Elections (EVT/WOTE)*, 2009.

[5] R. Anderson and R. Needham. Programming Satan's Computer. In Jan van Leeuwen, editor, *Computer Science Today: Recent Trends and Developments*, volume 1000 of *LNCS*, pages 426–440. Springer, 1995.

[6] M. Backes, C. Hritcu, and M. Maffei. Automated verification of remote electronic voting protocols in the applied pi-calculus. In *CSF'08: Proceed-*

ings of the 21st IEEE Computer Security Foundations Symposium, pages 195–209, Washington, USA, 2008. IEEE.

[7] A. Baskar, R. Ramanujam, and S. P. Suresh. Knowledge-based modelling of voting protocols. In *TARK'07: Proceedings of the 11th International Conference on Theoretical Aspects of Rationality and Knowledge*, pages 62–71, New York, USA, 2007. ACM.

[8] D. Bowen. Secretary of State Debra Bowen Moves to Strengthen Voter Confidence in Election Security Following Top-to-Bottom Review of Voting Systems. California Secretary of State, press release DB07:042 `http://www.sos.ca.gov/elections/voting_systems/ttbr/db07_042_ttbr_system_decisions_release.pdf`, August 2007.

[9] Bundesverfassungsgericht (Germany's Federal Constitutional Court). Use of voting computers in 2005 Bundestag election unconstitutional. Press release 19/2009 `http://www.bundesverfassungsgericht.de/en/press/bvg09-019en.html`, March 2009.

[10] D. Chaum, P. Y. A. Ryan, and S. Schneider. A practical, voter-verifiable election scheme. In *Proc. 10th European Symposium On Research In Computer Security (ESORICS'05)*, volume 3679 of *Lecture Notes in Computer Science*, pages 118–139. Springer, 2005.

[11] B. Chevallier-Mames, P.-A. Fouque, D. Pointcheval, J. Stern, and J. Traore. On Some Incompatible Properties of Voting Schemes. In *WOTE'06: Proceedings of the International Association for Voting Systems Sciences Workshop on Trustworthy Elections*, 2006.

[12] M. R. Clarkson, S. Chong, and A. C. Myers. Civitas: Toward a secure voting system. Technical Report 2007-2081, Cornell University, May 2007. Revised March 2008. `http://hdl.handle.net/1813/7875`.

[13] M. R. Clarkson, S. Chong, and A. C. Myers. Civitas: Toward a secure voting system. In *S&P'08: Proceedings of the 2008 IEEE Symposium on Security and Privacy*, pages 354–368, Washington, DC, USA, 2008. IEEE Computer Society.

[14] S. Delaune, S. Kremer, and M. D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 2009. To appear.

[15] A. Fujioka, T. Okamoto, and K. Ohta. A Practical Secret Voting Scheme for Large Scale Elections. In *ASIACRYPT'92: Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques*, pages 244–251, London, 1992. Springer.

[16] M. Jakobsson and A. Juels. Mix and match: Secure function evaluation via ciphertexts. In *ASIACRYPT '00: Proceedings of the 6th International*

Conference on the Theory and Application of Cryptology and Information Security, pages 162–177, London, UK, 2000. Springer.

[17] A. Juels, D. Catalano, and M. Jakobsson. Coercion-Resistant Electronic Elections. Cryptology ePrint Archive, Report 2002/165, 2002.

[18] A. Juels, D. Catalano, and M. Jakobsson. Coercion-resistant electronic elections. In *WPES '05: Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 61–70, New York, NY, USA, 2005. ACM. See also `http://www.rsa.com/rsalabs/node.asp?id=2860`.

[19] Ministerie van Binnenlandse Zaken en Koninkrijksrelaties (Netherland's Ministry of the Interior and Kingdom Relations). Stemmen met potlood en papier (Voting with pencil and paper). Press release `http://www.minbzk.nl/onderwerpen/grondwet-en/verkiezingen/nieuws--en/112441/stemmen-met-potlood`, May 2008.

[20] Participants of the Dagstuhl Conference on Frontiers of E-Voting. Dagstuhl accord. `http://www.dagstuhlaccord.org/`, 2007.

[21] B. Smyth, M. D. Ryan, S. Kremer, and M. Kourjieh. Towards automatic analysis of election verifiability properties. In *Joint Workshop on Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security (ARSPA-WITS'10)*, Lecture Notes in Computer Science. Springer, 2010. To appear.

[22] M. Talbi, B. Morin, V. V. T. Tong, A. Bouhoula, and M. Mejri. Specification of Electronic Voting Protocol Properties Using ADM Logic: FOO Case Study. In *ICICS'08: Proceedings of the 10th International Conference on Information and Communications Security Conference*, pages 403–418, London, 2008. Springer.

[23] UK Electoral Commission. Key issues and conclusions: May 2007 electoral pilot schemes. `http://www.electoralcommission.org.uk/elections/pilots/May2007`.