Automated Verification of Remote Electronic Voting Protocols in the Applied Pi-calculus

Michael Backes[1,2], Cătălin Hriţcu[1], and Matteo Maffei[1]
[1] Saarland University, Saarbrücken, Germany
[2] MPI-SWS
{backes,hritcu,maffei}@cs.uni-sb.de

Abstract

We present a general technique for modeling remote electronic voting protocols in the applied pi-calculus and for automatically verifying their security. In the first part of this paper, we provide novel definitions that address several important security properties. In particular, we propose a new formalization of coercion-resistance in terms of observational equivalence. In contrast to previous definitions in the symbolic model, our definition of coercion-resistance is suitable for automation and captures simulation and forced-abstention attacks. Additionally, we express inalterability, eligibility, and non-reusability as a correspondence property on traces. In the second part, we use ProVerif to illustrate the feasibility of our technique by providing the first automated security proof of the coercion-resistant protocol proposed by Juels, Catalano, and Jakobsson.

1. Introduction

Electronic voting is receiving increasing attention from governments, mass media, and the scientific community. On the one hand, electronic voting promises to simplify the voting procedure, to automate the count of votes, to guarantee the correctness of elections, and to prevent voter coercion. On the other hand, the errors in protocol design and the vulnerabilities in implementations [26, 10] raise considerable concerns about the reliability and safety of electronic voting systems [29, 21, 23]. This is not particularly surprising since designing security protocols has long been known to be error-prone. The distributed system aspects of multiple interleaved protocol runs render security analyses of such protocols awkward to make for humans. Formal methods, and in particular language-based techniques, e.g., [1, 2, 24, 14, 15, 7, 22], constitute salient tools for reliably analyzing security protocols. The main advantage of these techniques is automation: the human effort is limited to the specification of the protocol in a process calculus and the analysis is fully automated.

Coming up with a careful formalization and techniques for automated verification of electronic voting systems is arguably of paramount importance for the widespread acceptance of such systems in the scientific community, and hence might ultimately facilitate their successful deployment. While attention has been traditionally focused on the problem of supervised voting, where voters interact with a computing device under the supervision of election authorities, the more general and harder problem to solve is the problem of *remote voting*, where no supervision of voters or computing devices is in place [16].

1.1. Contributions

In this paper, we devise a general technique for modeling remote voting protocols in the applied pi-calculus [5] and for automatically analyzing their security properties. Our contributions can be summarized as follows.

First, we formalize three fundamental properties of electronic voting protocols, namely *inalterability* (votes are not modified), *eligibility* (only eligible voters can vote), and *non-reusability* (every voter can vote only once). To the best of our knowledge, this is the first formalization of these properties by means of correspondence assertions [31], in contrast to previous formalizations as control-flow properties [28].

Second, we devise a novel formalization of coercion-resistance for remote voting protocols. The formalization is given in terms of observational equivalence and is thus accessible to an automated analysis. Additionally, we formalize receipt-freeness and resistance to forced-abstention attacks in our setting, and we prove that under reasonable assumptions these properties are implied by coercion-resistance. Our formalizations are inspired by the seminal work of Delaune, Kremer, and Ryan [17, 19], which provides the first definitions in the symbolic model of vote-privacy, receipt-freeness, and coercion-resistance. In contrast to our formalization, however, their definitions of coercion-resistance are not amenable to automation since the one presented in [17] is based on a new notion of adaptive simulation, while the one later proposed in [19] uses a universal quantification over an infinite set of contexts. Furthermore, the former exhibits some undesirable properties, as the authors point out themselves [19], whereas the latter constrains the attacker to interact with the voter so that she casts a certain valid vote, i.e., the attacker is forced to essentially follow the intended protocol behavior. Additionally, these definitions do not consider forced-abstention attacks and do not apply to remote voting protocols.

Finally, we apply our technique to analyze the Juels, Catalano, and Jakobsson protocol [25]. The protocol is specified in the applied pi-calculus and the analysis is successfully conducted by ProVerif [12], for an unbounded number of honest and corrupted voters. This protocol is particularly important since it was the first protocol in the literature to satisfy a formal definition of coercion-resistance, it serves as the basis for the development of many modern election schemes for remote voting, e.g., [30, 27], and it is the protocol underlying the recently proposed Civitas system [16]. The Juels, Catalano, and Jakobsson protocol includes complex zero-knowledge proofs that so far have rendered an automated analysis of this protocol impossible. We solve the problem by exploiting a recently proposed technique [9] to express protocols based on zero-knowledge proofs in the applied pi-calculus and to analyze them using ProVerif.

1.2. Overview

Section 2 of the paper briefly reviews the applied pi-calculus and introduces the notation we use in the remainder of the paper. Section 3 explains how we model electronic voting protocols as applied pi-calculus processes. Section 4 presents the formalizations of several important security properties for remote voting protocols. In Section 5 we use our formal framework to model and analyze the security of the Juels, Catalano, and Jakobsson protocol using ProVerif. Section 6 concludes and provides directions for future work.

2. A Brief Review of the Applied Pi-calculus

We briefly recall the syntax and operational semantics of the applied pi-calculus from [5], and define the additional notation used in this paper. Terms are defined by means of a *signature* Σ, which is a set of function symbols, each with an arity. The set of terms is the term algebra built from names, variables, and function symbols in Σ applied to arguments. Let a, b, c range over channel names, and n, m over names of any sort. Also, let x, y, z range over variables and u range over both names and variables.

Terms are equipped with an *equational theory* E, i.e., an equivalence relation on terms that is closed under substitution of terms for variables and under application of term contexts (terms with a hole). We write $E \vdash M = N$ for equality and $E \not\vdash M = N$ and for inequality modulo E.

Plain processes are defined as follows. The null process **0** does nothing and is usually omitted from process specifications; $\nu n.P$ generates a fresh name n and then behaves as P; *if* $M = N$ *then* P *else* Q behaves as P if $E \vdash M = N$, and as Q otherwise; $a(x).P$ receives a message N from channel a and then behaves as $P\{N/x\}$; $\overline{a}\langle N\rangle.P$ outputs message N on channel a and then behaves as P; $P \mid Q$ executes P and Q in parallel; $!P$ generates an unbounded number of copies of P.

As usual, the scope of names and variables is delimited by restrictions and inputs. We write $\mathit{fv}(P)$ for the free variables and $\mathit{fn}(P)$ for the free names in a process P. We let $\widetilde{M}$ denote an arbitrary sequence $M_1, \ldots, M_k$ of terms, $\nu\widetilde{n}$ a sequence $\nu n_1 \ldots \nu n_k$ of name restrictions, and $\overline{c}\langle\widetilde{M}\rangle.P$ (resp. $c(\widetilde{x}).P$) the output (resp. input) of a tuple of terms, which can be encoded in the equational theory by the usual functions for pairs. Finally, for the sake of readability, we also use the processes $\mathit{let}\ x = M\ \mathit{in}\ P$ and $\mathit{let}\ x \in \widetilde{M}\ \mathit{in}\ P$, formally defined as $\nu a.(\overline{a}\langle M\rangle \mid a(x).P)$ and $\nu a.(\overline{a}\langle M_1\rangle \mid \ldots \mid \overline{a}\langle M_k\rangle \mid a(x).P)$, with $a \notin \mathit{fn}(P)$, respectively.

A *plain context* is a plain process with a hole. *Sequential contexts* are plain contexts that do not include replications and parallel compositions. If C is a context, then let $\mathsf{captured}(C)$ be the sequence of names and variables that are in scope for the hole, ordered according to their position in C. When this causes no confusion we will abbreviate the process $C[\mathbf{0}]$ as C.

As for the pi-calculus, the operational semantics of the applied-pi calculus is defined in terms of *structural equivalence* ($\equiv$) and *internal reduction* ($\rightarrow$). Structural equivalence captures rearrangements of parallel compositions and restrictions, and the equational rewriting of the terms in a process. Internal reduction defines the semantics of process synchronizations and conditionals. Additionally, *observational equivalence* ($\approx$) captures the equivalence of processes with respect to their dynamic behavior. Two processes are observationally equivalent if no context can distinguish them. For the formal definitions of these relations, we refer to Appendix A.2.

3. Formalizing Electronic Voting Protocols

We define an election process as an unbounded number of voters and trusted election authorities that are running in parallel and are sharing certain secrets. We distinguish three kinds of voters: *honest*, *corrupted*, and *ad-hoc*. Honest voters are issued an identity by a special authority and behave according to the protocol specification. In the setting considered in this paper, where the registration phase is trusted but the voting phase is not, corrupted voters will register and then simply output all their registration secrets on a public channel, so that the attacker can impersonate them in order to mount any sort of attack. Finally, ad-hoc voters can behave arbitrarily; they do not necessarily follow the protocol, but are also not necessarily corrupted. They have predefined public identities that allow one to track them across different instances of an election process. For this reason they are very useful when formalizing security properties as observational equivalences between election processes (e.g., we will use them to define coercion-resistance, receipt-freeness, etc.). We give the formal definition of election processes below, followed by additional explanations.

Definition 3.1 (Election Process) *An* election process *is a closed plain process*

$$EP \equiv \nu\widetilde{n}.(!V^{hon} \mid !V^{cor} \mid V_{id_1} \mid \ldots \mid V_{id_k} \mid ID \mid A_1 \mid \ldots \mid A_m)$$

such that

1. *there exist a private channel $c_{id} \in \widetilde{n}$ and two sequential contexts V^{reg} and V^{vote} such that $V^{hon} \equiv c_{id}(x_{id}).V^{reg}[let\ x_v \in \widetilde{v}\ in\ V^{vote}]$, $\mathsf{captured}(V^{reg}) \cap \mathsf{captured}(V^{vote}) = \emptyset$, and $\widetilde{v} \subseteq fn(EP)$. We define $\widetilde{v}$ to be the set of* valid votes*;*
2. *there exists a process V^c such that $V^{cor} \equiv c_{id}(x_{id}).V^c$;*
3. *there exists a process ID', and a public channel $c_{id\text{-}pub} \notin \widetilde{n}$ such that*

   $$ID \equiv (!\nu id.\overline{c_{id}}\langle id\rangle.\overline{c_{id\text{-}pub}}\langle id\rangle.let\ x_{id} = id\ in\ ID') \mid let\ x_{id} = id_1\ in\ ID' \mid \ldots \mid let\ x_{id} = id_k\ in\ ID';$$

   where for all $1 \leq i \leq k$ we have $id_i \notin \widetilde{n}$, for all $j \neq i$ we have $id_i \neq id_j$, and $c_{id} \notin fn(V^{reg}) \cup fn(V^{vote}) \cup fn(V^c) \cup (\bigcup_{i=1}^{k} fn(V_{id_i})) \cup fn(ID') \cup (\bigcup_{i=1}^{m} fn(A_i))$
4. *there exist $i \in [1, m]$, a public channel $c_{votes} \notin \widetilde{n}$, a variable x, a process P, and a context C such that $A_i \equiv C[\overline{c_{votes}}\langle x\rangle.P]$ and c_{votes} does not occur anywhere else in EP.*

The restricted names $\widetilde{n}$ model the secrets (e.g., some private channels) shared between the voters, denoted by V, and the election authorities, denoted by A.

An honest voter process is denoted by V^{hon}. It first receives an identity on the private channel c_{id}, registers, selects one of the valid vote choices non-deterministically, and then casts this vote (Condition 1). Corrupted voters receive an identity and are then under the control of the attacker (Condition 2). Since they are replicated, the number of honest and corrupted voters is not bounded. The ad-hoc voter processes V_{id_i} denote voters with a predefined identity id_i that do not necessarily follow the protocol, and are not replicated.

The *ID* process is an *identity issuer* that assigns to each voter a name that uniquely identifies her (Condition 3). These identities are public and hence known to attackers. They are used to make voter processes unique, and each voter holding a valid identity will be considered eligible in the election. Having public identities for voters is crucial when defining privacy properties, where the link between a vote and the identity of its originator has to be hidden, but both the votes and the identities of eligible voters have to be public. In addition, there exists some authority process A_i that is in charge of tallying the valid votes and outputting them on a special channel c_{votes} (Condition 4). These outputs constitute the result of the election.

Finally, we define an *election context S* as an election process with a hole that is in parallel composition with the voters.

4. Formalizing the Security Properties

This section devises novel formalizations in the applied pi-calculus for several important security properties of electronic-voting protocols. We start by defining a soundness property that entails inalterability, eligibility, and non-reusability. Then, we propose a novel formalization of coercion-resistance based on the standard notion of observational equivalence. We prove that, under reasonable assumptions, our notion of coercion-resistance implies immunity to forced-abstention attacks, vote-privacy and receipt-freeness.

4.1. Soundness

Correspondence assertions [31] are a natural way to formulate the soundness of an electronic voting protocol. The idea is to impose a causality relation among certain protocol events in execution traces. We refer the interested reader to [4] for the formal definition of events and execution traces in the applied pi-calculus. For our purpose, we annotate the election process *EP* with the following events: $\mathsf{newid}(id)$ is triggered by the identity issuer after giving the identity id to an eligible voter; $\mathsf{startid}(id)$ and $\mathsf{startcorid}(id)$ mark the start of the registration phase for an honest or corrupted voter id, respectively; $\mathsf{beginvote}(id, v)$ records the start of the voting phase for the honest voter id with the intention to vote for candidate v, while corrupted voters cast votes without asserting any event; $\mathsf{endvote}(v)$ indicates the tallying of vote v by the authority responsible for this. Ad-hoc voters may be annotated as corrupted or honest, depending on their behavior.

Definition 4.1 (Annotated Election Process) *An* annotated election process *is an election process*

$$EP \equiv \nu\widetilde{n}.(!V^{hon} \mid !V^{cor} \mid V_{id_1} \mid \ldots \mid V_{id_k} \mid ID \mid A_1 \mid \ldots \mid A_m).$$

annotated as follows:

1. $V^{hon} \equiv c_{id}(x_{id}).\mathsf{startid}(x_{id}).$
   $V^{reg}\,[let\ x_v \in \widetilde{v}\ in\ \mathsf{beginvote}(x_{id}, x_v).V^{vote}]$
   and neither V^{reg} nor V^{vote} contain any event;

2. $V^{cor} \equiv c_{id}(x_{id}).\mathsf{startcorid}(x_{id}).V^c$ *where V^c does not contain any event;*

3. *for each i we have that either $V_{id_i} \equiv \mathsf{startid}(id_i).V_i'$ with V_i' containing at most one* $\mathsf{beginvote}(id_i, v)$ *event for some v, or $V_{id_i} \equiv \mathsf{startcorid}(id_i).V_i'$ where V_i' does not contain any event;*

4. *ID* $\equiv$ $(!\nu id.\mathsf{newid}(id).\overline{c_{id}}\langle id\rangle.\overline{c_{\textit{id-public}}}\langle id\rangle.$
   $let\ x_{id} = id\ in\ ID') \mid$
   $\mathsf{newid}(id_1).let\ x_{id} = id_1\ in\ ID' \mid \ldots \mid$
   $\mathsf{newid}(id_k).let\ x_{id} = id_k\ in\ ID'$
   where $\mathsf{newid}(\cdot)$ *does not occur anywhere else in EP, and ID′ does not contain any event;*

5. $A_i \equiv C[\mathsf{endvote}(x).\overline{c_{votes}}\langle x\rangle.P]$ *and C, P and A_j for $j \neq i$ do not contain any event.*

In the literature, soundness is typically defined as three separate properties: *inalterability* (no one can change anyone else's vote), *eligibility* (only eligible voters are able to vote) and *non-reusability* (every voter can vote only once). We define a single notion of soundness that encompasses these three properties.

Definition 4.2 (Soundness) *A trace t guarantees* soundness *if and only if the following conditions hold:*

1. *for any t_1, t_2, v such that $t = t_1 :: \mathsf{endvote}(v) :: t_2$, there exist id, t', t'', t''' such that*
   - *(a)* $t_1 = t' :: \mathsf{startid}(id) :: t'' :: \mathsf{beginvote}(id, v) :: t'''$ *and* $t' :: t'' :: t''' :: t_2$ *guarantees soundness;*
   - *(b) or* $t_1 = t' :: \mathsf{startcorid}(id) :: t''$*, and* $t' :: t'' :: t_2$ *guarantees soundness.*

2. *for any t_1, t_2, id such that $t = t_1 :: \mathsf{startid}(id) :: t_2$ or $t = t_1 :: \mathsf{startcorid}(id) :: t_2$, the events* $\mathsf{startid}(id)$ *and* $\mathsf{startcorid}(id)$ *do not occur in $t_1 :: t_2$.*

3. *for any t_1, t_2, id such that $t = t_1 :: \mathsf{startid}(id) :: t_2$ or $t = t_1 :: \mathsf{startcorid}(id) :: t_2$, the event* $\mathsf{newid}(id)$ *occurs in t_1.*

An annotated election process EP guarantees soundness if and only if all its possible traces guarantee soundness.

Inalterability is modeled by requiring that every counted vote matches a vote cast by some voter, either honest (Condition 1a) or corrupted (Condition 1b). Non-reusability is modeled by requiring that the matching between the events $\mathsf{endvote}(v)$ and $\mathsf{beginvote}(id, v)$ is injective. Notice that the structure of election processes described in Definition 4.2 guarantees that the events $\mathsf{startid}(id)$ and $\mathsf{startcorid}(id)$ depend on distinct ids (Condition 2). Similarly, Condition 3, which captures eligibility, is enforced syntactically by the shape of the process. Condition 1 can be checked automatically by ProVerif.

4.2. Coercion-resistance

As described in [25], coercion-resistance captures four different properties:

Receipt-freeness. A coercer cannot force a voter to cast a certain vote and to provide a receipt that would certify her vote.

Immunity to simulation attacks. A voter cannot be forced into providing all the secrets required for a coercer to impersonate her, since there is no way for the coercer to tell the difference between real and fake secrets.

Immunity to forced-abstention attacks. A coercer should not be able to tell whether a particular voter has voted or not, so that he cannot force the voter to abstain.

Immunity to randomization attacks. A voter cannot be forced to divulge or nullify her vote by using random messages received from the coercer.

We define coercion-resistance as immunity to simulation attacks and later prove that this definition implies immunity to forced-abstention attacks and receipt-freeness. Although our definition considers an arbitrary attacker, we do not formally address randomization attacks in this paper and leave this topic as future work. Our definition is based on observational equivalence and is similar in spirit to the cryptographic definition proposed in [25].

We call a voting protocol coercion-resistant if an attacker cannot distinguish a coerced voter providing him with secret material and abstaining from voting, from a voter providing him with fake secrets and actively participating in the vote. In the first case, we assume that the coercer can effectively mount a simulation attack and impersonate the voter, and for example cast whatever vote he wants on her behalf, or abstain from voting. Since we reason about remote voting protocols, we assume that the process V^{vote}, by which an honest voter casts his vote, uses *public* channels.

We first define a voter process $V_i^{coerced(c)}$ that complies with the demands of the coercer, so it takes part in the registration phase, forwards all generated or received secrets to the coercer on channel c, and then abstains from voting:

$$V_i^{coerced(c)} \equiv let\ x_{id} = i\ in\ V^{reg}\,[\overline{c}\langle\widetilde{u}\rangle]\,,$$

where $\widetilde{u} = \mathsf{captured}(V^{reg})$. If a protocol is coercion-resistant, then there exists a strategy for the voter to fake registration secrets and cheat the coercer: this strategy is modeled as a plain context V^{fake}. This context has to satisfy two simple well-formedness conditions: $\mathsf{captured}(V^{reg}) \subseteq \mathsf{captured}(V^{fake})$ and $V^{reg}\left[V^{fake}\left[\mathbf{0}\right]\right] \approx V^{reg}\left[\mathbf{0}\right]$, which we implicitly assume satisfied in the following. To satisfy the first condition, bound names and variables can be reassigned by the νn and $let\ x = M\ in$ constructs, respectively. These conditions allow us to replace the registration secrets by the fake ones and use the context $V^{reg}[V^{fake}]$ in place of V^{reg}.

As mentioned before, the context V^{fake} models the strategy used to cheat a coercer, so it is of course dependent on the particular election protocol being analyzed. The process $V_i^{cheat(c)}(v')$ registers and votes as a normal voter, but cheats a coercer by providing him with fake secrets.

$$\begin{aligned} V_i^{cheat(c)}(v') \equiv\ & let\ x_{id} = i\ in \\ & V^{reg}[let\ x_v = v'\ in\ V^{vote} \mid V^{fake}\left[\overline{c}\langle\widetilde{u}\rangle\right]], \end{aligned}$$
where $\widetilde{u} = \mathsf{captured}(V^{reg})$.

Intuitively, an election context S is coercion-resistant if $S[V_i^{coerced(c)}]$ is observationally equivalent to $S[V_i^{cheat(c)}(v')]$, i.e., the two processes are indistinguishable for any attacker. However, this does not hold, since in the first case the coerced voter abstains, while in the second it casts a vote. The coercer can thus distinguish the two election processes since both the *votes* published in the final tally and the *number of messages* exchanged in the voting phase are different.

In order to compensate for the vote of the voter cheating the coercer, we add one more voting process V_j on each side of the observational equivalence. The voter that complies with the demands of the coercer and abstains will run in parallel with a voting process that votes v', while the voting process that cheats the coercer and casts a vote v' will run in parallel with an abstaining voter. This models a scenario where the adversary is not certain about the behavior of at least one additional voter. We define a voter $V_j(v)$ having identity j and casting a valid vote v as

$$V_j(v) \equiv let\ x_{id} = j\ in\ V^{reg}\left[let\ x_v = v\ in\ V^{vote}\right].$$

Additionally, we formalize the behavior of a voter V_j^{abs} participating in the registration phase and then abstaining from voting as follows:

$$V_j^{abs} \equiv let\ x_{id} = j\ in\ V^{reg}\left[\mathbf{0}\right].$$

Therefore, we could try to define coercion-resistance in terms of the following observational equivalence:

$$S[\ V_i^{coerced(c)} \mid V_j(v')\ ] \approx S[\ V_i^{cheat(c)}(v') \mid V_j^{abs}\ ].$$

Even though the number of messages exchanged in the voting phase is now the same and the vote of V_i is compensated by the additional voter V_j, observational equivalence still does not hold. On the left-hand side, the coercer gets hold of real registration secrets that he can use to cast a valid vote, while on the right-hand side the coercer gets fake registration secrets, so any vote he will cast using them will be invalid. If we assumed that the coercer would cast a fixed valid vote v, then we could balance both sides of the equivalence by adding an additional voter k, which on one side casts a vote with fake registration secrets while on the other casts the vote v:

$$\begin{gathered} S[\ V_i^{coerced(c)} \mid V_j(v') \mid V_k^{inv\text{-}reg}\ ] \\ \approx \\ S[\ V_i^{cheat(c)}(v') \mid V_j^{abs} \mid V_k(v?)\ ] \end{gathered} \tag{1}$$

where $V_k^{inv\text{-}reg}$ is defined as follows:

$$V_k^{inv\text{-}reg} = let\ x_{id} = k\ in\ V^{reg}[V^{fake}[let\ x_v \in \widetilde{v}\ inV^{vote}]].$$

Intuitively, this models a scenario where the coercer does not know the exact distribution of votes and cannot tell whether the third voter casts an invalid ballot or a valid vote that balances the outcome of the election processes. Note that in all coercion-resistant protocols (e.g., [25, 6, 30, 27]) casting a ballot with invalid registration secrets is guaranteed to nullify the vote inside, without an eavesdropper being able to tell any difference other than the final result of the tally. This is not the case with other ways of nullifying the vote such as casting an invalid vote, since the ballot may contain a proof that the vote inside is valid [25].

In any case, the vote the coercer chooses to cast is not known beforehand, in general. Even more, the coercer could just abstain, or try to vote more than once. Since the coercer is modeled as an arbitrary context, balancing his vote is not easy at all. In [17], Delaune, Kremer and Ryan define a new notion of adaptive simulation in an attempt to solve this problem. However, we want to use the standard notion of observational equivalence in order to take advantage of the automation support in ProVerif.

We now examine the equivalence (1) more carefully. In case of duplicate votes it is up to the tallying authority to decide which one of these votes to consider in the tally. So identifying the relevant vote cast by the coercer on behalf of V_i requires the collaboration of both V_i and the tallying authority. However, in most election protocols the tallying authority processes the votes only in a special tallying phase, when new votes are no longer accepted. So even if we have a way to find out the vote of the coercer after duplicate elimination, this will happen most likely too late for the voter k to be able to cast another vote that would balance the result of the tally.

Even if we ignored the problem and assumed a way to identify the vote of the coercer early enough so that k can still cast a ballot, equivalence (1) would still not hold until certain synchronization problems are solved. In particular, on the left-hand side the vote cast by the coercer can be processed even if no other voter performs visible actions, while in the right-hand side at least the balancing voter k has to participate in the voting phase until the vote of the coercer can be output. This means that the replicated instance of the tallying authority processing the coercer's vote needs to synchronize with the instance processing k's vote, which would further complicate the framework.

In order to keep things both simple and general we do not model the third voter explicitly, but replace it by a process which "extracts" the vote the coercer casts on behalf of V_i and tallies it directly. This extractor registers as k, gets the same (real or fake) secrets received by the coercer from the coerced voter V_i, receives from the tallying authority all the votes considered after duplicate elimination and identifies the vote cast by the coercer on behalf of V_i. On the right-hand side the extractor outputs the vote cast by the coercer on channel c_{votes}, the special channel on which the result of the tally is published, while on the left-hand side the extractor does not output anything, thus abstracting voter k casting a vote with invalid registration secrets. Letting the extractor directly balance the coercer's vote is an abstraction that considerably simplifies the structure of the election process and the definition of coercion-resistance.

The extractor is dependent on the construction of the particular electronic voting protocol and has to be provided by the user. In order to achieve its goal, the extractor has additionally access to the secrets of the election authorities so that it can distinguish the vote of the coercer from the other votes. Since it has access to all this information, we need to ensure that it cannot leak it and can only use it in the way described above. We hence impose syntactic restrictions on the shape of the extractor, which will later be strengthened with semantic restrictions on its behaviour.

Definition 4.3 (Extractor) *A context $E_k^{c_1,c_2,z}$ is an* extractor *if and only if*

$$E_k^{c_1,c_2,z} = \ \mathit{let}\ x_{id} = k\ \mathit{in}\ V^{reg}[\nu\widetilde{m}.(c_1(x).P_1 \mid !c_2(y).P_2 \mid C\,[\mathit{if}\ z \in \widetilde{v}\ \mathit{then}\ [\,]])]$$

for some plain processes P_1, P_2 and a sequential context C such that $c_1, c_2 \notin fn(P_1) \cup fn(P_2) \cup fn(C)$, $z \in \mathsf{captured}(C)$, all inputs and outputs in P_1, P_2, and C occur on the private channels in $\widetilde{m}$, and such channels are never output.

The channels c_1 and c_2 are the channels shared by the extractor with the coerced voter and the tallying authority, respectively. If the coercer casts a vote, then the variable z should hold this vote. The context C is required to be sequential so it does not contain any replications, which means that $E_k^{c_1,c_2,z}\ [\overline{c_{votes}}\langle z\rangle]$ can tally at most one vote.

For convenience we define the process $V_i^{coerced(c,c_1)}$ that complies with the demands of the coercer but outputs its secrets on two channels c and c_1, the first for the coercer and the second for the extractor.

$$V_i^{coerced(c,c_1)} \equiv \mathit{let}\ x_{id} = i\ \mathit{in}\ V^{reg}\,[\overline{c}\langle\widetilde{u}\rangle \mid \overline{c_1}\langle\widetilde{u}\rangle]$$

where $\widetilde{u} = \mathsf{captured}(V^{reg})$. The process $V_i^{cheat(c,c_1)}(v)$ is defined from $V_i^{cheat(c)}(v)$ in a very similar way.

We can now express coercion-resistance as the observational equivalence between the following election processes:

$$\begin{gathered} S'[\ V_i^{coerced(c,c_1)} \mid V_j(v') \mid E_k^{c_1,c_2,z}\,[\mathbf{0}]\ ] \\ \approx \\ S'[\ V_i^{cheat(c,c_1)}(v') \mid V_j^{abs} \mid E_k^{c_1,c_2,z}\,[\overline{c_{votes}}\langle z\rangle]\ ] \end{gathered}$$

The first process contains the voter V_i that complies with the demands of the coercer, running in parallel with the voter V_j casting a vote v', and the process $E_k^{c_1,c_2,z}\,[\mathbf{0}]$, that is intuitively equivalent to a voter nullifying her vote. In the second election process the voter V_i cheats the coercer by providing him with fake registration secrets and then votes v', the voter V_j participates in the registration phase and then abstains, and the extractor process $E_k^{c_1,c_2,z}\,[\overline{c_{votes}}\langle z\rangle]$ tallies the vote the coercer casts on behalf of V_i.

Definition 4.4 (Coercion-resistance) *An election context S guarantees* coercion-resistance *if there exist channels c, c_1, and c_2, a sequential process V^{fake}, an extractor $E_k^{c_1,c_2,z}$, and an election context S', such that*

1. *there exist an election context S'' and two authority processes A, A' such that $S \equiv S''[A \mid [\,]]$, $S' \equiv \nu c_1, c_2.S''[A' \mid [\,]]$, and $\nu c_2.(A' \mid !c_2(x)) \approx A$;*
2. $S'[\ V_i^{coerced(c,c_1)} \mid V_j(v') \mid E_k^{c_1,c_2,z}\,[\mathbf{0}]\ ]$ $\approx\ S'[\ V_i^{cheat(c,c_1)}(v') \mid V_j^{abs} \mid E_k^{c_1,c_2,z}\,[\overline{c_{votes}}\langle z\rangle]\ ]$ *where $v' \in \widetilde{v}$ is a valid vote;*
3. $\nu c.S'[!c(x) \mid V_i^{cheat(c,c_1)}(v') \mid V_j^{abs} \mid E_k^{c_1,c_2,z}\,[\overline{c_{votes}}\langle z\rangle]]$ $\approx\ S[\ V_i(v') \mid V_j^{abs} \mid V_k^{abs}\ ]$;
4. *Let $P = c(\widetilde{x}).\mathit{let}\ x_v = v\ \mathit{in}\ V^{vote}\{\widetilde{x}/\widetilde{u}\}$, $v \in \widetilde{v}$, $\widetilde{u} = \mathsf{captured}(V^{reg})$, and $\widetilde{x} \cap \widetilde{u} = \emptyset$ then*
   $\nu c.S'[P \mid V_i^{cheat(c,c_1)}(v') \mid V_j^{abs} \mid E_k^{c_1,c_2,z}\,[\overline{c_{votes}}\langle z\rangle]] \approx$ $\nu c.S'[P \mid V_i^{cheat(c,c_1)}(v') \mid V_j^{abs} \mid E_k^{c_1,c_2,z}\,[\overline{c_{votes}}\langle v\rangle]]$;
5. $S[V_i^{inv\text{-}reg}] \approx \nu c_{votes}.(!c_{votes}(x) \mid S[V_i(v)])$, *where v is a valid vote.*

The definition of coercion-resistance uses a modified election context S' that only differs from S in that the tallying authority additionally outputs messages on the channel c_2 shared with the extractor (Condition 1). The main equivalence in the definition was already given and discussed (Condition 2). We additionally impose two restrictions that characterize the intended behaviour of the extractor process $E_k^{c_1,c_2,z}\,[\overline{c_{votes}}\langle z\rangle]$. If the cheated coercer abstains, then the extractor needs to abstain as well (Condition 3); and if the cheated coercer casts a valid vote using the fake registration secrets he received from V_i, then the extractor needs to tally precisely this vote (Condition 4). Finally, we pose an additional restriction that justifies the abstraction of the third voter by the extractor (Condition 5): votes with invalid registration secrets are silently discarded by the tallying authority. If this was not the case a coercer could easily distinguish real from fake registration secrets.

There are two reasons why this definition of coercion-resistance is suitable for our purpose. First, it only uses the standard notion of observational-equivalence, and no universal quantification over processes or contexts, which makes it suitable for automation. Second, this definition is strong, as it considers an arbitrary attacker and an unbounded number of honest and corrupted participants, and it captures immunity to simulation and forced-abstention attacks, vote-privacy and receipt-freeness. In the next two subsections we study these other properties formally.

4.3. Immunity to Forced-abstention Attacks and Vote-privacy

We call an election process immune to forced-abstention attacks if no attacker is able to distinguish a voter that casts a vote from a voter that abstains. In order to balance the votes and the number of messages sent on the network, we define this property using two different voters: one that casts a vote and one that abstains. The protocol is immune to forced-abstention attacks if the attacker is not able to tell which one of the two voters actually casts the vote.

Definition 4.5 (Immunity to Forced-abstention Attacks) *An election context S is* immune to forced-abstention attacks *if for a valid vote v we have that*

$$S\left[V_i(v) \mid V_j^{abs}\right] \approx S\left[V_i^{abs} \mid V_j(v)\right].$$

In [17] an election protocol is defined to guarantee vote-privacy if an attacker is not able to distinguish between a process in which two voters cast one vote each, from the same process where these two votes are swapped.

Definition 4.6 (Vote-privacy) *An election context S guarantees* vote-privacy *if for two valid votes v and v' we have*

$$S\left[V_i(v) \mid V_j(v')\right] \approx S\left[V_i(v') \mid V_j(v)\right].$$

We show that, under a very reasonable assumption – there exists at least one additional abstaining voter, coercion-resistance implies immunity to forced-abstention attacks, which in turn implies vote-privacy.

Theorem 4.7 *If S guarantees coercion-resistance, then S guarantees immunity to forced-abstention attacks, assuming that there is at least one additional abstaining voter, i.e.,*

$$S[\,V_i^{abs} \mid V_j(v') \mid V_k^{abs}\,] \approx S[\,V_i(v') \mid V_j^{abs} \mid V_k^{abs}\,].$$

Theorem 4.8 *Immunity to forced-abstention attacks implies vote-privacy, assuming that there is at least one additional abstaining voter, i.e.,*

$$S\left[V_i(v) \mid V_j(v') \mid V_k^{abs}\right] \approx S\left[V_i(v') \mid V_j(v) \mid V_k^{abs}\right].$$

4.4. Receipt-freeness

Intuitively, a protocol guarantees receipt-freeness if a voter does not gain any information that can be used to prove to a coercer that she voted in a certain way. This definition thus refers to an attacker that does not try to vote by impersonating the coerced voter, as in coercion-resistance, but just tries to get a proof that the voter voted in a certain way. In [17] receipt-freeness is defined as follows:

Definition 4.9 (DKR-Receipt-freeness) *An election context S is* receipt-free *if there exists a closed plain process V' such that*

1. $\nu c.!c(x) \mid V' \approx V_i(v')$
2. $S\left[V_i^c(v) \mid V_j(v')\right] \approx S\left[V' \mid V_j(v)\right]$

The process $V_i^c(v)$ behaves as a regular voting process, with the exception that all the secrets are revealed on a public channel c as soon as they are generated or received from private channels. This definition is only suitable for protocols where the votes are cast on private channels, which are set up in advance and never leaked. In the setting studied in this paper, namely remote voting protocols, the channels used to cast the votes are public. If a voter reveals her registration secrets, then an attacker can effectively impersonate her, and cast a vote on her behalf. So the second condition clearly fails for any protocol we would consider, since the vote chosen dynamically by an attacker is not statically predictable. This is also the reason why the definition of coercion-resistance proposed in [19] does not apply to remote voting protocols.

We model receipt-freeness in our setting by letting the coerced voter reveal all her secrets only after the voting phase. This is consistent with the literature on electronic voting [11], and intuitively provides weaker capabilities to an attacker than in the case of coercion-resistance, where the secrets are revealed right after the registration phase and can be used by the attacker to impersonate the coerced voter. While this solution is conceptually elegant, it requires us to extend the applied pi-calculus with primitives modeling *protocol phases*. Following [3], we extend the syntax of processes with the process $t : P$: intuitively, t is a number model-ling a global synchronization clock and $t : P$ is a process behaving as P at phase t and getting stuck in the other phases. For more detail on the semantics of phases, we refer to Appendix A.3. We remark that the phase command is supported by ProVerif.

In the remainder of this section, we only consider election processes where each output on channel c_{votes} made by the tallying authority or the extractor is of the form $t_{\mathit{votes}} : \overline{c_{\mathit{votes}}}\langle M \rangle$, for some fixed phase t_{votes} that intuitively corresponds to the election result publication phase. For instance the extractor is now defined as:

$$\begin{aligned} E_k^{c_1,c_2,z} = \;\; & \mathit{let}\ x_{id} = k\ \mathit{in}\ V^{\mathit{reg}}[\nu\widetilde{m}.(c_1(x).P_1 \mid !c_2(y).P_2 \\ & \qquad \mid C\,[\mathit{if}\ z \in \widetilde{v}\ \mathit{then}\ t_{\mathit{votes}} : [\,]])] \end{aligned}$$

A voter that provides a receipt to a coercer first registers, then votes v as requested by the coercer, and finally, once the voting phase is over, reveals all the secrets she generated or received. Formally, the process $V_i^{\mathit{receipt}(c)}(v)$ voting v and outputting a receipt on channel c is defined as follows:

$$\begin{aligned} V_i^{\mathit{receipt}(c)}(v) \equiv \;& \mathit{let}\ x_{id} = i\ \mathit{in} \\ & V^{\mathit{reg}}[\mathit{let}\ x_v = v\ \mathit{in}\ V^{\mathit{vote}}[t_{\mathit{votes}} : \overline{c}\langle\widetilde{u}\rangle]], \\ \text{where } \widetilde{u} = \;& \mathsf{captured}(V^{\mathit{vote}}) \cup \mathsf{captured}(V^{\mathit{reg}}) \end{aligned}$$

We could try to define receipt-freeness as in Definition 4.9, where in the second condition we just replace $V_i^c(v)$ by $V_i^{\mathit{receipt}(c)}(v)$. However, such a definition would be too restrictive, since the voter cheating the coercer has to provide a fake receipt but otherwise has to act *exactly* like an honest voter.

This rules out the following generic strategy to provide a fake receipt in coercion-resistant protocols. The voter V' registers, casts her real vote, and in parallel she generates fake secrets, casts the vote the coercer asked for using these fake secrets, and finally provides the receipt of this invalid voting. Intuitively, if there was a way for a receipt-freeness attacker to tell whether this receipt is fake or not, then we could construct a coercion-resistance attacker that would be able to tell whether some registration secrets are fake or not. This coercion-resistance attacker would first get the registration secrets from a coerced voter, use them to cast a vote and obtain a receipt, then forward this receipt to the receipt-freeness attacker. If the secrets he received are fake, the coercion-resistance attacker we constructed is basically simulating a voter cheating a receipt-freeness attacker, which by our assumption would be able to tell whether the receipt is fake or not. So by using the receipt-freeness attacker as an oracle, we could build an attacker against coercion-resistance.

In our opinion this is a valid strategy for providing fake receipts in a coercion-resistant protocol. However, such a strategy is not captured by Definition 4.9 because in order to avoid coercion the voter has to cast two votes, one of which is invalid, thus violating the first condition. We therefore devise a definition tailored towards this particular strategy of providing fake receipts.

Definition 4.10 (Receipt-freeness for Remote Voting) *An election context S is* receipt-free *if there exists a plain process V' such that*

1. $\nu c.(!c(x) \mid V') \approx \mathit{let}\ x_{id} = i\ \mathit{in}$
   $V^{\mathit{reg}}\left[\mathit{let}\ x_v = v'\ \mathit{in}\ V^{\mathit{vote}} | V^{\mathit{fake}}[\mathit{let}\ x_v \in \widetilde{v}\ \mathit{in}\ V^{\mathit{vote}}]\right]$

2. $S[\, V_i^{\mathit{receipt}(c)}(v) \mid V_j(v') \mid V_k^{\mathit{inv\text{-}reg}} \,]$
   $\approx S[\, V' \mid V_j(v) \mid V_k^{\mathit{abs}} \,]$

The main difference with respect to Definition 4.9 is that the voter V' does not only vote v' as a regular voter, but additionally uses V^{fake} to generate fake secrets, casts an extra vote using them, and provides a receipt of this invalid voting (Condition 1). In order to balance this additional noise we add an additional voter k that votes with fake registration secrets in case the voter i complies with the request of the coercer (left-hand side of Condition 2), and simply abstains if i cheats the coercer by casting a vote with fake secrets (right-hand-side of Condition 2).

We can finally state that coercion-resistance implies receipt freeness up to the abstraction of the third voter by the extractor, as inherited from the definition of coercion-resistance.

Theorem 4.11 *If S is an election context that guarantees coercion-resistance then there exists V' such that*

1. $\nu c.(!c(x) \mid V')$
   $\approx \mathit{let}\ x_{id} = i\ \mathit{in}\ V^{\mathit{reg}}[\mathit{let}\ x_v = v'\ \mathit{in}\ V^{\mathit{vote}} \mid$
   $V^{\mathit{fake}}[\mathit{let}\ x_v \in \widetilde{v}\ \mathit{in}\ V^{\mathit{vote}} \mid \overline{c_1}\langle\widetilde{u}\rangle]]$

2. $S[\, V_i^{\mathit{receipt}(c)}(v) \mid V_j(v') \mid V_k^{\mathit{abs}} \,] \approx$
   $S'[\, V' \mid V_j^{\mathit{abs}} \mid E_k^{c_1,c_2,z}\,[\overline{c_{\mathit{votes}}}\langle v \rangle]\,]$

where $\widetilde{u} = \mathsf{captured}(V^{\mathit{reg}})$

5. Analysis of the Juels, Catalano, and Jakobsson Protocol

This section presents the first analysis in the formal model of the Juels, Catalano, and Jakobsson protocol [25]. The importance of this protocol is twofold: it was the first protocol in the literature to satisfy a formal definition of coercion-resistance, and it laid the basis for the development of many modern election schemes for remote voting (e.g., [30, 27] and the recently proposed Civitas [16]).

The protocol involves a *registrar* in charge of issuing secret credentials to voters, a set of *tallying authorities* responsible for processing ballots, jointly counting votes, and publishing the final tally, a set of *voters*, and a *bulletin board*. A threshold encryption systems guarantees the safety of the protocol even if a minority of the tallying authorities is corrupted.

The protocol is divided into three phases: registration, voting, and tallying. In the registration phase, voters receive a credential from the registrar. This credential constitutes a proof of eligibility, which is then used in the voting phase. Additionally, the registrar publishes on the digitally signed bulletin board the credential encrypted by the tallying authority's public key. The protocol assumes the registrar to be trustworthy and the channel between the registrar and the voter to be untappable. The registration phase is depicted below:

$$\begin{tikzcd}
\textit{Voter} & \textit{Registrar} \arrow[r, "\{cred\}_{\mathsf{pk}(kT)}"] \arrow[l, "cred"] & \textit{BB}
\end{tikzcd}$$

In the voting phase, voters cast their vote on an anonymous public channel. Voters output on the channel the vote and the credential encrypted by the tallying authority's public key and a zero-knowledge proof of knowledge of the credential and of validity of the vote. The protocol assumes a fixed number of candidates and the proof guarantees that the vote is for one of these candidates.

$$\begin{tikzcd}
\textit{Voter} \arrow[r, "{\{cred\}_{\mathsf{pk}(kT)},\{vote\}_{\mathsf{pk}(kT)},\mathsf{ZK}}"] & \textit{Tallier}
\end{tikzcd}$$

Finally, the tallying authorities check the proofs, eliminate duplicates, check the credentials and eventually publish the set of valid votes. A plaintext equivalence test (PET) between the encrypted credential received from the voter and the encrypted credential read from the digitally signed bulletin board allows the tallying authorities to jointly check the validity of the credentials without decrypting them, thus preserving their secrecy.

$$\begin{tikzcd}
\textit{Tallier} \arrow[r, "votes"] & \textit{BB}
\end{tikzcd}$$

5.1. Equational Theory

The base equational theory considered in this paper includes function symbols for constructing and destructing pairs, encrypting and decrypting messages using asymmetric cryptography, signing messages, verifying signatures, and performing PETs. The binary functions eq, $\wedge$, and $\vee$ model equality test, conjunction, and disjunction, respectively, thus allowing for expressing monotone Boolean formulas. For the sake of readability, we shall often use infix notation for eq, $\wedge$, and $\vee$ and replace eq by $=$.

$$\begin{array}{lcl}
\mathsf{eq}(x,x) & = & \mathsf{true}\\
\wedge(\mathsf{true},\mathsf{true}) & = & \mathsf{true}\\
\vee(\mathsf{true},x) & = & \mathsf{true}\\
\vee(x,\mathsf{true}) & = & \mathsf{true}\\
\mathsf{pet}(\mathsf{enc}(x,\mathsf{pk}(y),z),\mathsf{enc}(x,\mathsf{pk}(y),w),\mathsf{sk}(y)) & = & \mathsf{true}\\
\mathsf{fst}(\mathsf{pair}(x,y)) & = & x\\
\mathsf{snd}(\mathsf{pair}(x,y)) & = & y\\
\mathsf{dec}(\mathsf{enc}(x,\mathsf{pk}(y),z),\mathsf{sk}(y)) & = & x\\
\mathsf{msg}(\mathsf{sign}(x,\mathsf{sk}(y))) & = & x\\
\mathsf{ver}(\mathsf{sign}(x,\mathsf{sk}(y)),x,\mathsf{pk}(y)) & = & \mathsf{true}
\end{array}$$

The equational theory also contains the function symbols and equational rules recently introduced in [9] for abstractly reasoning about non-interactive zero-knowledge proofs in the applied pi-calculus. A non-interactive zero-knowledge proof is represented as a term of the form $\mathsf{ZK}_{i,j}(\widetilde{M},\widetilde{N},F)$. The statement preserves the secrecy of the terms $\widetilde{M}$, called the statement's *private component*, while the terms $\widetilde{N}$, called the statement's *public component*, are revealed to the verifier and to the adversary. The formula F constitutes a term without names and variables, which is built upon distinguished nullary functions α_i and β_i with $i \in \mathbb{N}$. Hence $\mathsf{ZK}_{i,j}$ is a function of arity $i+j+1$, but we shall often omit arities and write this statement as $\mathsf{ZK}(\widetilde{M};\widetilde{N};F)$, letting semicolons separate the respective components.

Definition 5.1 **((i,j)-formulas)** *We call a term an (i,j)-formula if the term contains neither names nor variables, and if for every α_k and β_l occurring therein, we have $k \in [1,i]$ and $l \in [1,j]$.*

The values α_i and β_j in F constitute placeholders for the terms M_i and N_j, respectively. For instance, the term

$$\mathsf{ZK}(\ \mathsf{sk}(k)\ ;\ m,\mathsf{pk}(k)\ ;\ \beta_1 = \mathsf{dec}(\mathsf{enc}(\beta_1,\beta_2),\alpha_1)\ )$$

denotes a zero-knowledge proof of knowledge of the secret key $\mathsf{sk}(k)$ corresponding to the public key $\mathsf{pk}(k)$. More precisely, the statement reads: "There exists a secret key such that the decryption of the ciphertext $\mathsf{enc}(m,\mathsf{pk}(k))$ with such a key yields m". As mentioned before, m and $\mathsf{pk}(k)$ are revealed by the proof while $\mathsf{sk}(k)$ is kept secret. This is formalized in general terms by the following infinite set of equational rules:

$$\begin{array}{lcll}
\mathsf{Public}_p(\mathsf{ZK}_{i,j}(\widetilde{M},\widetilde{N},F)) & = & N_p & \text{with } p \in [1,j]\\
\mathsf{Formula}(\mathsf{ZK}_{i,j}(\widetilde{M},\widetilde{N},F)) & = & F &
\end{array}$$

where Public_p and $\mathsf{Formula}$ constitute functions of arity 1. Since there is no destructor associated to the statement's private component, the terms $\widetilde{M}$ are kept secret. We define a statement $\mathsf{ZK}_{i,j}(\widetilde{M},\widetilde{N},F)$ to hold if F is an (i,j)-formula and the formula obtained by substituting all α_k's and β_l's in F with the corresponding values M_k and N_l is true. Verification of a statement $\mathsf{ZK}_{i,j}$ with respect to a formula is modeled as a function $\mathsf{Ver}_{i,j}$ of arity 2 that is defined by the following equational rule:

$$\begin{array}{ll} \mathsf{Ver}_{i,j}(F, \mathsf{ZK}_{i,j}(\widetilde{M}, \widetilde{N}, F)) = \mathsf{true} & \text{iff} \\ \quad 1) \quad E \vdash F\{\widetilde{M}/\widetilde{\alpha}\}\{\widetilde{N}/\widetilde{\beta}\} = \mathsf{true} & \\ \quad 2) \quad F \text{ is an } (i,j)\text{-formula} & \end{array}$$

where $\{\widetilde{M}/\widetilde{\alpha}\}\{\widetilde{N}/\widetilde{\beta}\}$ denotes the substitution of each α_k with M_k and of each β_l with N_l. This rule guarantees in the abstract model the *soundness* and *correctness* of zero-knowledge protocols. As shown in [9], we can compile this infinite equational theory into a finite and equivalent one, which is suitable for automated analysis by ProVerif.

5.2. Protocol Specification

The protocol specification in the applied pi-calculus is reported in Table 1. Process voter models honest voters: they receive an identifier from process identityissuer, receive a credential on the private channel *chVR* shared between voters and the registrar, choose a vote, and output a zero-knowledge proof conveying the encrypted credential and the encrypted vote. The statement

$$\begin{array}{l} \mathsf{enc}(\alpha_3, \beta_3, \alpha_4) = \beta_1 \wedge \\ \mathsf{enc}(\alpha_1, \beta_3, \alpha_2) = \beta_2 \wedge \\ (\alpha_3 = \beta_4 \vee \alpha_3 = \beta_5 \vee \alpha_3 = \beta_6) \end{array}$$

of the zero-knowledge proof says that the first and second public component are a credential and a vote, both are encrypted by the tallying authority's public key and the vote is one among v_A, v_B, and v_C. For the sake of simplicity, we consider an election with three candidates. Process corvoter models corrupted participants leaking their credentials. Process identityissuer generates a new identifier, which is then sent to the voter and to the registrar. Process registrar receives an identifier, generates a credential, sends the credential to the voter, signs and publishes the encrypted credential, and finally sends the encrypted credential to the tallying authority on an internal channel. This corresponds to the tallying authority reading the encrypted credentials from the bulletin board.

In our model, we consider a single tallying authority and abstract the threshold encryption scheme by the standard equational theory for asymmetric cryptography: as a consequence, we assume the trustworthiness of the tallying authority. Process tallier receives a ballot which is a zero-knowledge proof, checks its validity, and, if the plaintext equivalence test between the encrypted credential in the ballot and one of the encrypted credentials on the bulletin board succeeds[1], decrypts and publishes the vote. We remark that each encrypted credential is processed by the tallying authority only once and this guarantees the non-reusability property.

5.3. Security Analysis

We first verified the soundness property stated in Definition 4.2 (Condition 1). This required us to annotate the processes as specified in Table 1. The analysis was performed by ProVerif and succeeded. As a result, the Juels, Catalano, and Jakobsson Protocol is proved to guarantee inalterability, eligibility, and non-reusability for an unbounded number of honest voters and an unbounded number of corrupted participants. Notice that non-reusabity crucially relies on certain messages being sent only once on internal channels (e.g., the credential sent on channel *chRT*). To guarantee this, our implementation in ProVerif uses nonce handshakes. This does not affect the observable semantics of the system, but is necessary to counter the over-approximation of the static analysis.

The analysis of coercion-resistance relies on the additional processes reported in Table 2, namely the extractor and the three processes for the coerced, cheating, and abstaining voter. Notice that the faking strategy of the cheating voter consists in generating a fresh credential and sending it to the coercer. Finally, the modified tallying authority sends the encrypted credential and the encrypted vote received from the network together with the encrypted credential received from the registrar on the channel c_2 shared with the extractor.

We verified Definition 4.4 using ProVerif. The tool supports observational equivalence proofs expressed as biprocesses [13]: as an example, $\nu n.\overline{a}\langle n\rangle \approx \nu n.\overline{a}\langle h(n)\rangle$, where h is free in the equational theory, is written $\nu n.\overline{a}\langle \mathsf{choice}[n, h(n)]\rangle$. Therefore the proof is automated, but some human effort is still required to transform each equivalence of Definition 4.4 into a biprocess. The transformation is mostly straightforward and the only interesting condition is 2, which requires that $\mathsf{JCJ-CR1} \approx \mathsf{JCJ-CR2}$ (see Table 2).

Understanding why this condition holds is crucial for expressing the equivalence in the form of a biprocess. The two sides of the equivalence differ because of the two voters and the extractor, as defined in Definition 4.4. At run-time, each of them is associated to a replicated instance of the tallying authority in charge of processing their vote, say T_1 for the coerced voter i, T_2 for the additional voter j, and T_3 for the extractor k. However, the equivalence between the two sides is not straightforward since in the left-hand side the vote v' chosen by j is processed by T_2, while in the right-hand side, it is processed by T_1. Similarly, the vote cast by the

[1]In our model the PET is used just for consistency with the original protocol specification. This cryptographic primitive is more meaningful in a setting where the threshold encryption system is modeled explicitly.

Table 1 Juels, Catalano, and Jakobsson Protocol in the Applied Pi-calculus

$$
\begin{array}{llll}
\mathsf{voter} \triangleq & \mathsf{corvoter} \triangleq & \mathsf{identityissuer} \triangleq & \mathsf{registrar} \triangleq \\
\quad c_{id}(id). & \quad c_{id}(id). & \quad \nu id. & \quad chIR(id). \\
\quad \mathsf{startid}(id). & \quad \mathsf{startcorid}(id). & \quad \mathsf{newid}(id). & \quad \nu cred. \\
\quad chVR(cred). & \quad \overline{chVR}(cred). & \quad \overline{c_{id}}\langle id\rangle. & \quad \nu r. \\
\quad \mathit{let}\ vote \in \{v_A, v_B, v_C\}\ \mathit{in} & \quad \overline{pub}\langle cred\rangle & \quad \overline{chIR}\langle id\rangle. & \quad \overline{chVR}\langle cred\rangle. \\
\quad \mathsf{beginvote}(id, vote). & & \quad \overline{pub}\langle id\rangle & \quad \overline{pub}\langle \mathsf{sign}(\mathsf{enc}(cred, pk(kT), r), sk(kR))\rangle. \\
\quad \nu r_1.\nu r_2. & & & \quad \overline{chRT}\langle \mathsf{enc}(cred, pk(kT), r)\rangle \\
\quad \overline{pub}\langle zk\rangle & & &
\end{array}
$$

$$
\begin{array}{l}
\mathsf{tallier} \triangleq \\
\quad pub(zkp). \\
\quad \mathit{if}\ \mathsf{Ver}_{4,6}(proof, zkp)\ \mathit{then} \\
\quad \mathit{if}\ \mathsf{Public}_4(proof) = v_a\ \mathit{then} \\
\quad \mathit{if}\ \mathsf{Public}_5(proof) = v_b\ \mathit{then} \\
\quad \mathit{if}\ \mathsf{Public}_6(proof) = v_c\ \mathit{then} \\
\quad \mathit{let}\ encvote = \mathsf{Public}_1(zkp)\ \mathit{in} \\
\quad \mathit{let}\ enccred = \mathsf{Public}_2(zkp)\ \mathit{in} \\
\quad chRT(enccred_1). \\
\quad \mathit{if}\ \mathsf{pet}(enccred, enccred_1, sk(kT))\ \mathit{then} \\
\quad \mathit{let}\ vote = \mathsf{dec}(encvote, sk(kT))\ \mathit{in} \\
\quad \overline{c_{votes}}\langle vote\rangle.
\end{array}
$$

$$
\begin{array}{rl}
\mathsf{JCJ} \triangleq & \nu c_{id}.\nu chIR.\nu chVR.\nu chRT.\nu kT.\nu kR.\overline{pub}\langle \mathsf{pair}(pk(kT), pk(kR))\rangle. \\
& (!\mathsf{voter} \mid !\mathsf{corvoter} \mid !\mathsf{identityissuer} \mid !\mathsf{registrar} \mid !\mathsf{tallier}) \\
zk = & \mathsf{ZK}_{4,6}(\ cred, r_2, vote, r_1; \\
& \qquad \mathsf{enc}(vote, \mathsf{pk}(kT), r_1), \mathsf{enc}(cred, \mathsf{pk}(kT), r_2), \mathsf{pk}(kT), v_a, v_b, v_c; \\
& \qquad proof\,) \\
proof = & \mathsf{enc}(\alpha_1, \beta_3, \alpha_2) = \beta_2 \wedge \\
& \mathsf{enc}(\alpha_3, \beta_3, \alpha_4) = \beta_1 \wedge \\
& (\alpha_3 = \beta_4 \vee \alpha_3 = \beta_5 \vee \alpha_3 = \beta_6)
\end{array}
$$

coercer is processed by T_1 on the left-hand side, while it is processed by T_3 on the right-hand side: remember that the tallying authority is responsible for passing the encrypted vote received from the network to the extractor. This scenario is depicted below, where $T_i(x)$ means that the tallying authority T_i processes credential x:

$$\begin{tikzcd}
T_1(cr_i) & T_2(cr_j) & T_3(cr_k) \\
{} \arrow[u, "v?"] & {} \arrow[u, "v'"] & \\
& \approx & \\
T_1(cr_i) & T_2(cr_j) & T_3(cr_k) \\
{} \arrow[u, "v'"] & & {} \arrow[u, "v?"]
\end{tikzcd}$$

The analysis done by ProVerif relies on an over-approximation of the process semantics that enforces the same synchronizations on the two sides of the equivalence. For instance, if the vote of the coercer is received by T_1 in the left-hand side, then the same holds in the right-hand side. Furthermore, each process in the left-hand side has to simulate the behavior of its syntactic counterpart in the right-hand side (and vice-versa): for example the equivalence $\nu n.\overline{a}\langle m\rangle|\overline{a}\langle h(n)\rangle \approx \nu n.\overline{a}\langle n\rangle|\overline{a}\langle m\rangle$ holds, but the proof of $\nu n.\overline{a}\langle \mathsf{choice}[m, n]\rangle \mid \overline{a}\langle \mathsf{choice}[h(n), m]\rangle$ does not succeed in ProVerif. The solution is to replace the left- or the right-hand side by a structurally equivalent process, which corresponds to encoding the proof strategy in the biprocess. For instance, if we swap the two processes on the right-hand side of the previous equivalence, we obtain the biprocess $\nu n.\overline{a}\langle \mathsf{choice}[m, m]\rangle \mid \overline{a}\langle \mathsf{choice}[h(n), n]\rangle$ and now the proof succeeds. In our case, we have to explicitly encode in the biprocess the proof strategy described before, which requires to swap the credentials processed by each tallying authority:

$$\begin{tikzcd}
T_1(\mathsf{choice}[cr_i, cr_k]) & T_2(\mathsf{choice}[cr_j, cr_i]) & T_3(\mathsf{choice}[cr_k, cr_j]) \\
{} \arrow[u, "v?"] & {} \arrow[u, "v'"] &
\end{tikzcd}$$

For more details on the ProVerif specification of the Juels, Catalano, and Jakobsson protocol and on the analysis thereof, we refer the interested reader to [8].

6. Conclusions

In this paper we present a general technique for modeling remote voting protocols in the applied pi-calculus and for automatically verifying their security for an unbounded number of honest and corrupted voters. In particular, we give a new definition of coercion-resistance in terms of observational equivalence. This definition captures immunity to simulation and forced-abstention attacks, vote-privacy and receipt-freeness, and is suitable for automation. In addition, we formalize inalterability, eligibility, and non-reusability as a correspondence property on traces, which can also be analyzed automatically. We illustrate the proposed theoretical framework by modeling and analyzing the Juels, Catalano, and Jakobsson protocol using ProVerif.

This paper essentially reduces the problem of verifying coercion-resistance to checking observational equivalence. This property can be automatically verified by ProVerif, but

Table 2 Juels, Catalano, and Jakobsson Protocol: Processes for Coercion-resistance

$$\begin{array}{llll}
\mathsf{coercedvoter} \triangleq & \mathsf{cheatingvoter} \triangleq & \mathsf{absvoter} \triangleq & \mathsf{tallierE} \triangleq \\
\quad c_{id}(id). & \quad c_{id}(id). & \quad c_{id}(id). & \quad \mathit{pub}(\mathit{zkp}). \\
\quad \mathit{chVR}(\mathit{cred}). & \quad \mathit{chVR}(\mathit{cred}). & \quad \mathit{chVR}(\mathit{cred}) & \quad \mathit{if}\ \mathsf{Ver}_{4,6}(\mathit{proofenc}, \mathit{zkp})\ \mathit{then} \\
\quad \overline{c}\langle \mathit{cred}\rangle. & \quad \nu \mathit{fakecred}. & & \quad \mathit{if}\ \mathsf{Public}_4(\mathit{proof}) = v_a\ \mathit{then} \\
\quad \overline{c_1}\langle \mathit{cred}\rangle & \quad \overline{c}\langle \mathit{fakecred}\rangle \mid & & \quad \mathit{if}\ \mathsf{Public}_5(\mathit{proof}) = v_b\ \mathit{then} \\
 & \quad \mathit{let\ vote} = v_A\ \mathit{in} & & \quad \mathit{if}\ \mathsf{Public}_6(\mathit{proof}) = v_c\ \mathit{then} \\
 & \quad \nu r_1.\nu r_2. & & \quad \mathit{let\ encvote} = \mathsf{Public}_1(\mathit{zkp})\ \mathit{in} \\
 & \quad \overline{\mathit{pub}}\langle \mathit{zk}\rangle & & \quad \mathit{let\ enccred} = \mathsf{Public}_2(\mathit{zkp})\ \mathit{in} \\
 & & & \quad \mathit{chRT}(\mathit{enccred}_1). \\
 & & & \quad \overline{c_2}\langle(\mathit{enccred}, \mathit{enccred}_1, \mathit{encvote})\rangle. \\
 & & & \quad \mathit{if}\ \mathsf{pet}(\mathit{enccred}, \mathit{enccred}_1, sk(kT))\ \mathit{then} \\
 & & & \quad \mathit{let\ vote} = \mathsf{dec}(\mathit{encvote}, sk(kT))\ \mathit{in} \\
 & & & \quad \overline{c_{\mathit{votes}}}\langle \mathit{vote}\rangle.
\end{array}$$

$$\begin{array}{l}
\mathsf{extractor}[\,] \triangleq \\
\quad (c_{id}(id).\mathit{chVR}(\mathit{credext}). \\
\quad\ \ \nu a.\nu b. \\
\quad (c_1(\mathit{fakecred}).!\overline{a}\langle \mathit{fakecred}\rangle) \mid \\
\quad (!c_2((\mathit{enccred}, \mathit{enccred}_1, \mathit{encvote})). \\
\quad\ \ a(\mathit{fakecred}). \\
\quad\ \ \mathit{let\ vote} = \mathsf{dec}(\mathit{encvote}, \mathsf{sk}(kT))\ \mathit{in} \\
\quad\ \ \mathit{let\ cred} = \mathsf{dec}(\mathit{enccred}, \mathsf{sk}(kT))\ \mathit{in} \\
\quad\ \ \mathit{let\ cred}_1 = \mathsf{dec}(\mathit{enccred}_1, \mathsf{sk}(kT))\ \mathit{in} \\
\quad\ \ \mathit{if\ cred} = \mathit{fakecred\ then} \\
\quad\ \ \mathit{if\ cred}_1 = \mathit{credext\ then} \\
\quad\ \ \overline{b}\langle \mathit{vote}\rangle) \mid \\
\quad (b(z).\mathit{if}\ z \in \{v_a, v_b, v_c\}\ \mathit{then} \\
\quad\ \ [\,])) \ \%\ \text{either 0 or } \overline{c_{\mathit{votes}}}\langle z\rangle
\end{array}$$

$$\begin{array}{ll}
\mathsf{JCJ{-}CR1} \triangleq & \nu c_1.\nu c_2.\nu c_{id}.\nu \mathit{chIR}.\nu \mathit{chVR}.\nu \mathit{chRT}.\nu \mathit{kT}.\nu \mathit{kR}. \\
 & \overline{\mathit{pub}}\langle \mathsf{pair}(\mathsf{pk}(\mathit{kT}), \mathsf{pk}(\mathit{kR}))\rangle. \\
 & (!\mathsf{voter} \mid !\mathsf{corvoter} \mid !\mathsf{identityissuer} \mid !\mathsf{registrar} \mid !\mathsf{tallierE} \mid \\
 & \mathsf{coercedvoter} \mid \mathsf{voter}(v_A) \mid \mathsf{extractor}(0)) \\
\mathsf{JCJ{-}CR2} \triangleq & \nu c_1.\nu c_2.\nu c_{id}.\nu \mathit{chIR}.\nu \mathit{chVR}.\nu \mathit{chRT}.\nu \mathit{kT}.\nu \mathit{kR}. \\
 & \overline{\mathit{pub}}\langle \mathsf{pair}(\mathsf{pk}(\mathit{kT}), \mathsf{pk}(\mathit{kR}))\rangle. \\
 & (!\mathsf{voter} \mid !\mathsf{corvoter} \mid !\mathsf{identityissuer} \mid !\mathsf{registrar} \mid !\mathsf{tallierE} \mid \\
 & \mathsf{cheatingvoter} \mid \mathsf{voterabs} \mid \mathsf{extractor}(\overline{c_{\mathit{votes}}}\langle z\rangle))
\end{array}$$

this still requires non-negligible human effort to transform process specifications into biprocesses. Extending the scope of ProVerif to a wider class of process equivalences is subject of active research [20]. Notice, however, that our approach is not tailored to a specific tool and could in principle rely on other techniques, such as the one based on symbolic bisimulation that has been recently proposed in [18].

As future work, we plan to analyze other protocols for remote voting, such as [6, 30, 27], and the protocol underlying the recently proposed Civitas system [16]. It would also be interesting to formalize in the symbolic model other interesting security properties such as immunity to randomization attacks, individual and universal verifiability, completeness, and resilience to denial-of-service attacks.

Acknowledgments We thank the anonymous reviewers for their valuable comments and Catherine Howell for her insightful remarks.

References

[1] M. Abadi. Secrecy by typing in security protocols. *Journal of the ACM*, 46(5):749–786, 1999.

[2] M. Abadi and B. Blanchet. Secrecy types for asymmetric communication. In *Proc. 4th International Conference on Foundations of Software Science and Computation Structures (FOSSACS)*, volume 2030 of *Lecture Notes in Computer Science*, pages 25–41. Springer-Verlag, 2001.

[3] M. Abadi, B. Blanchet, and C. Fournet. Automated verification of selected equivalences for security protocols, 2007. To appear in Journal of Logic Algebric Programming.

[4] M. Abadi, B. Blanchet, and C. Fournet. Just fast keying in the pi calculus. *ACM Transactions on Information and System Security*, 10(3):9, 2007.

[5] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proc. 28th Symposium on Principles of Programming Languages (POPL)*, pages 104–115. ACM Press, 2001.

[6] A. Acquisti. Receipt-free homomorphic elections and write-in ballots. Cryptology ePrint Archive, Report 2004/105, 2004. `http://eprint.iacr.org/`.

[7] M. Backes, A. Cortesi, and M. Maffei. Causality-based abstraction of multiplicity in cryptographic protocols. In *Proc. 20th IEEE Symposium on Computer Security Foundations (CSF)*, pages 355–369. IEEE Computer Society Press, 2007.

[8] M. Backes, C. Hriţcu, and M. Maffei. ProVerif scripts for the Juels, Catalano, and Jakobsson protocol. Available at: `www.infsec.cs.uni-sb.de/~maffei/ev.zip`.

[9] M. Backes, M. Maffei, and D. Unruh. Zero-knowledge in the applied pi-calculus and automated verification of the direct anonymous attestation protocol. To appear in IEEE Symposium on Security and Privacy, 2008.

[10] J. Bannet, D. W. Price, A. Rudys, J. Singer, and D. S. Wallach. Hack-a-vote: Security issues with electronic voting systems. *IEEE Security & Privacy*, 2(1):32–37, 2004.

[11] J. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *Proc. 26th Annual ACM Symposium on Theory of Computing (STOC)*, pages 544–553. ACM Press, 1994.

[12] B. Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In *Proc. 14th IEEE Computer Security Foundations Workshop (CSFW)*, pages 82–96. IEEE Computer Society Press, 2001.

[13] B. Blanchet, M. Abadi, and C. Fournet. Automated verification of selected equivalences for security protocols. In *Proc. 20th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 331–340. IEEE Computer Society Press, 2005.

[14] C. Bodei, M. Buchholtz, P. Degano, F. Nielson, and H. R. Nielson. Static validation of security protocols. *Journal of Computer Security*, 13(3):347–390, 2005.

[15] M. Bugliesi, R. Focardi, and M. Maffei. Dynamic types for authentication. *Journal of Computer Security*, 15(6):563–617, 2007.

[16] M. R. Clarkson, S. Chong, and A. C. Myers. Civitas: A secure voting system. To appear in IEEE Symposium on Security and Privacy, 2008.

[17] S. Delaune, S. Kremer, and M. Ryan. Coercion-resistance and receipt-freeness in electronic voting. In *Proc. 19th IEEE Computer Security Foundations Workshop (CSFW)*, pages 28–42. IEEE Computer Society Press, 2006.

[18] S. Delaune, S. Kremer, and M. Ryan. Symbolic bisimulation for the applied pi calculus. In *In Proc. of Foundations of Software Technology and Theoretical Computer Science, 27th International Conference (FSTTCS 2007)*, Lecture Notes in Computer Science, pages 133–145. Springer-Verlag, 2007.

[19] S. Delaune, S. Kremer, and M. Ryan. Verifying privacy-type properties of electronic voting protocols. Research Report LSV-08-01, Laboratoire Spécification et Vérification, ENS Cachan, France, 2008.

[20] S. Delaune, M. Ryan, and B. Smyth. Automatic verification of privacy properties in the applied pi calculus. To appear in 2nd Joint iTrust and PST Conferences on Privacy, Trust Management and Security (IFIPTM'08), 2008.

[21] D. L. Dill, B. Schneier, and B. Simons. Voting and technology: who gets to count your vote? *Communications of the ACM*, 46(8):29–31, 2003.

[22] S. Doghmi, J. Guttman, and F. Thayer. Searching for shapes in cryptographic protocols. In *Proc. 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, Lecture Notes in Computer Science, pages 523–538. Springer-Verlag, 2007.

[23] D. Evans and N. Paul. Election security: Perception and reality. *IEEE Security & Privacy*, 2(1):24–31, 2004.

[24] A. D. Gordon and A. Jeffrey. Types and effects for asymmetric cryptographic protocols. *Journal of Computer Security*, 12(3):435–484, 2004.

[25] A. Juels, D. Catalano, and M. Jakobsson. Coercion-resistant electronic elections. In *Proc. 4nd ACM Workshop on Privacy in the Electronic Society (WPES)*, pages 61–70. ACM Press, 2005.

[26] T. Kohno, A. Stubblefield, A. D. Rubin, and D. S. Wallach. Analysis of an electronic voting system. In *Proc. 25th IEEE Symposium on Security & Privacy*, pages 27–42. IEEE Computer Society Press, 2004.

[27] T. Krivoruchko. Robust coercion-resistant registration for remote e-voting. IAVoSS Workshop On Trustworthy Elections (WOTE 2007), June 2007. Available at: `http://research.microsoft.com/conferences/WOTE2007/papers/09.pdf`.

[28] C. R. Nielsen, E. H. Andersen, and H. R. Nielson. Static validation of a voting protocol. In *Automated Reasoning for Security Protocol Analysis (ARSPA 2005)*, volume 135 of *Electronic Notes on Theoretical*

Table 3 Syntax of the Applied Pi-calculus

Terms

M, N	$::=$	a, b, c, i, j, k, n, m	*names*
		x, y, z	*variables*
		$\mathsf{f}(M_1, \ldots, M_k)$	*functions*

where $\mathsf{f} \in \Sigma$ and k is the arity of f.

Plain Processes

P, Q	$::=$	$\mathbf{0}$	*nil*
		$\nu n.P$	*name restriction*
		$\mathit{if}\, M = N \mathit{\ then\ } P \mathit{\ else\ } Q$	*conditional*
		$u(x).P$	*input*
		$\overline{u}\langle N\rangle.P$	*output*
		$P \mid Q$	*parallel*
		$!P$	*replication*

Extended Processes

A, B	$::=$	P	*plain process*
		$A \mid B$	*parallel*
		$\nu u.A$	*restriction*
		$\{M/x\}$	*active substitution*

Computer Science, pages 115–134. Elsevier Science Publishers Ltd., 2005.

[29] A. D. Rubin. Security considerations for remote electronic voting. *Communications of the ACM*, 45(12):39–44, 2002.

[30] S. G. Weber, R. Araujo, and J. Buchmann. On coercion-resistant electronic elections with linear work. In *ARES '07: Proceedings of the The Second International Conference on Availability, Reliability and Security*, pages 908–916. IEEE Computer Society Press, 2007.

[31] T. Y. C. Woo and S. S. Lam. A semantic model for authentication protocols. In *Proc. 14th IEEE Symposium on Security & Privacy*, pages 178–194. IEEE Computer Society Press, 1993.

A. Review of the Applied Pi-Calculus

A.1 Syntax

The complete syntax of the applied pi-calculus [5] is given in Table 3. In addition to what is presented in Section 2 we define extended processes, frames, and evaluation contexts.

Extended processes consists of plain processes, parallel compositions, restrictions and *active substitutions* $\{M/x\}$, i.e., floating substitutions that may apply to any process that they come into contact with. To control the scope of an active substitution $\{M/x\}$, we can restrict the variable x. Intuitively, $\nu x.(P \mid \{M/x\})$ restricts the scope of the substitution $\{M/x\}$ to process P and has the same semantics as a let construct. If the variable x is not restricted, as it is the case in the process $(P \mid \{M/x\})$, then the substitution is exported by the process and the environment has immediate access to M.

A context C closes A if $C[A]$ is closed (i.e., it does not contain free variables). An *evaluation context* $\mathcal{E}$ is a context of the following form:

$$\mathcal{E} ::= [\,] \mid \nu u.\mathcal{E} \mid (\mathcal{E} \mid A) \mid (A \mid \mathcal{E}).$$

A *frame* is an extended process built up from **0** and active substitutions by parallel composition and restriction. We let ϕ and ψ range over frames. The domain $\mathit{dom}(\phi)$ of a frame ϕ is the set of variables that ϕ exports, i.e., those variables x for which ϕ contains a substitution $\{M/x\}$ not under a restriction on x. Every extended process A can be mapped to a frame $\phi(A)$ by replacing every plain process embedded in A with **0**. 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 for A's dynamic behavior.

Table 4 Structural Equivalence

PAR-0	$A \equiv A \mid 0$
PAR-A	$A_1 \mid (A_2 \mid A_3) \equiv (A_1 \mid A_2) \mid A_3$
PAR-C	$A_1 \mid A_2 \equiv A_2 \mid A_1$
REPL	$!P \equiv P \mid !P$
RES-0	$\nu n.0 \equiv 0$
RES-C	$\nu u.\nu u'.A \equiv \nu u'.\nu u.A$
RES-PAR	$A_1 \mid \nu u.A_2 \equiv \nu u.(A_1 \mid A_2), u \notin \mathit{fv}(A_1) \cup \mathit{fn}(A_1)$
ALIAS	$\nu x.\{M/x\} \equiv 0$
SUBST	$\{M/x\} \mid A \equiv \{M/x\} \mid A\{M/x\}$
REWRITE	$\{M/x\} \equiv \{N/x\}, \quad E \vdash M = N$

A.2 Semantics

The operational semantics of the applied-pi calculus is defined in terms of *structural equivalence* ($\equiv$) and *internal reduction* ($\rightarrow$). Structural equivalence captures rearrangements of parallel compositions, restrictions and active substitutions, and the equational rewriting of the terms in a process.

Definition A.1 (Structural Equivalence) *Structural equivalence ($\equiv$) is the smallest equivalence relation on extended processes that satisfies the rules in Table 4 and that is closed under α-renaming of names and variables, and under application of evaluation contexts.*

Internal reduction defines the semantics of the communication primitives and conditionals.

Table 5 Internal reduction

Comm
$$\overline{a}\langle x\rangle.P \mid a(x).Q \rightarrow\ P \mid Q$$

Then
$$\mathit{if}\, M = M\ \mathit{then}\ P\ \mathit{else}\ Q\ \rightarrow\ P$$

Else
$$\frac{E \not\vdash M = N \qquad M, N \text{ ground}}{\mathit{if}\, M = N\ \mathit{then}\ P\ \mathit{else}\ Q\ \rightarrow\ Q}$$

Definition A.2 (Internal Reduction) *Internal reduction ($\rightarrow$) is the smallest relation on extended processes that satisfies the rules in Table 5 and that is closed under structural equivalence and under application of evaluation contexts.*

We write $A \Downarrow a$ to denote that A can send a message on a, i.e., $A \rightarrow^* \mathcal{E}[\overline{a}\langle M\rangle.P]$ for some evaluation context $\mathcal{E}$ that does not bind a.

We write $A \Downarrow a$ to denote that A can send a message on a, i.e., $A \rightarrow^* \mathcal{E}[\overline{a}\langle M\rangle.P]$ for some evaluation context $\mathcal{E}$ that does not bind a.

Definition A.3 (Observational Equivalence) *Observational equivalence ($\approx$) is the largest symmetric relation $\mathcal{R}$ between closed extended processes with the same domain such that $A\mathcal{R}B$ implies:*

1. *if $A \Downarrow a$, then $B \Downarrow a$;*
2. *if $A \rightarrow^* A'$, then $B \rightarrow^* B'$ and $A'\mathcal{R}B'$ for some B';*
3. *$\mathcal{E}[A]\mathcal{R}\mathcal{E}[B]$ for all closing evaluation contexts $\mathcal{E}$.*

A.3 Protocol Phases

Following [3], we extend the syntax of processes with the form $t : P$, where the phase prefix t is a number and the process P can only contain numbers bigger than t. Intuitively, this models a global synchronization clock where P is active only at phase t. The structural equivalence relation is extended as follows:

$$\begin{array}{lcll} \nu a.t : P & \equiv & t : \nu a.P & \\ t : (P \mid Q) & \equiv & t : P \mid t : Q & \\ t : t' : P & \equiv & t' : P & \text{if } t \leq t' \end{array}$$

Internal reduction at phase t ($\rightarrow_t$) is the smallest relation on extended processes closed under structural equivalence and application of evaluation contexts that satisfies the following rule:

$$P \rightarrow Q \quad \Rightarrow \quad t : P \rightarrow_t t : Q$$

All definitions now use $\rightarrow_{\_} = \bigcup_{t \geq 0} \rightarrow_t$ instead of $\rightarrow$. We also introduce the notion of observational equivalence up to phase t, namely the observational equivalence relation restricted to barbs at phase less equal than t.

Definition A.4 (Obs. Equivalence Up to Phase t) *Observational equivalence up to phase t, written $\approx_t$, is the largest symmetric relation $\mathcal{R}$ between closed extended processes with the same domain such that $A\mathcal{R}B$ implies:*

1. *if $A \Downarrow_t a$, then $B \Downarrow_t a$;*
2. *if $A \rightarrow^* A'$, then $B \rightarrow^* B'$ and $A'\mathcal{R}B'$ for some B';*
3. *$C[A]\mathcal{R}C[B]$ for all closing evaluation contexts C;*

where $A \Downarrow_t a$ if and only if $A \rightarrow^* C[t' : \overline{a}\langle M\rangle.P]$ for some evaluation context C that does not bind a and some $t' \leq t$.

The following proposition states a monotonicity condition for observational equivalence.

Proposition A.5 (Monotonicity of Obs. Equivalence) *For any $t' \geq t$ we have that*

$$\approx\ \subseteq\ \approx_{t'}\ \subseteq\ \approx_t .$$