Election verifiability in electronic voting protocols* (Preliminary version**)

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

[1] School of Computer Science, University of Birmingham, UK
[2] LSV, ENS Cachan & CNRS & INRIA, France
[3] Université de Toulouse, France

research@bensmyth.com, M.D.Ryan@cs.bham.ac.uk,
kremer@lsv.ens-cachan.fr, kourjieh@irit.fr

Abstract. We present a symbolic definition of election verifiability for electronic voting protocols. Our definition is given in terms of reachability assertions in the applied pi calculus and is amenable to automated reasoning using the tool ProVerif. The definition distinguishes three aspects of verifiability, which we call individual, universal, and eligibility verifiability. It also allows us to determine precisely what aspects of the system are required to be trusted. We demonstrate our formalism by analysing the protocols due to Fujioka, Okamoto & Ohta and Juels, Catalano & Jakobsson; the latter of which has been implemented by Clarkson, Chong & Myers.

Key words: Electronic voting protocols, election verifiability, applied pi calculus, ProVerif, automated reasoning.

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. However, current deployment has shown that the security benefits are very hard to realise [12, 20, 11, 22]. Those systems rely on the trustworthiness of the servers and software that is used to collect, tally and count the votes, and on the individuals that manage those servers. In practice, it is very hard to establish the required level of trust.

The concept of *election verifiability* that has emerged in the academic literature [16, 19] aims to address this problem. It significantly reduces the necessity to trust electronic systems, by allowing voters and election observers to verify independently that votes have been recorded, tallied and counted correctly. To emphasise a voter's ability

* This work has been partly supported by the EPSRC projects *UbiVal* (EP/D076625/2), *Trustworthy Voting Systems* (EP/G02684X/1) & *Verifying Interoperability Requirements in Pervasive Systems* (EP/F033540/1), the *ANR SeSur AVOTÉ* project and the University of Birmingham's *Blue BEAR* cluster.

** A revised version of this paper will be available from the following address in due course: http://www.bensmyth.com/publications/09wissec/

to verify the results of the entire election process, it is sometimes called *end-to-end* verifiability.

We define election verifiability in a formal and general setting, and we analyse several voting protocols from the literature. We work in the applied pi calculus [2], and where possible we use the ProVerif [10] tool to automate the verification. The calculus and the tool have already been successful in analysing other properties of voting systems [14, 6].

In this paper we use the term *bulletin board* to refer to the output produced at the end of an election process. This will include, at least, legitimate ballots (also called bulletin board entries); and the election outcome (a multiset of votes). We remark that illegitimate ballots are assumed to be discarded prior to the end of the election process.

Election verifiability allows voters and observers to verify that the election outcome corresponds to the votes legitimately cast. We distinguish three 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; and, a voter can check that her own vote is included.
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; and, a voter can check that her own vote is considered legitimate.

(Note that some authors use the term "universal verifiability" to refer to the conjunction of what we call "universal verifiability" and "eligibility verifiability". This distinction is made for compatibility with protocols which do not offer eligibility verifiability.) These three aspects of verifiability are related to the following *correctness properties* [6], defined with respect to honest protocol executions:

Inalterability: no one can change a voter's vote.
Declared result: the election outcome is the correct sum of the votes cast.
Eligibility: only registered voters can vote and at most once.

Election verifiability properties are intuitively stronger than correctness properties, since they assert that voters and observers can check that the correctness properties hold, event when administrators deviate from the protocol.

We define election verifiability in terms of a test which can be performed on the bulletin board by either a voter or an observer. The test succeeds if and only if the election outcome corresponds to ballots legitimately cast. That is, the following criteria must be satisfied: 1) all votes cast by voters are included in the bulletin board (possibly subject to some re-vote weeding policy); 2) the election outcome corresponds to the votes cast; and 3) each vote included in the election outcome was cast by a unique registered voter.

1.1 Contribution

We present the first formal definition of election verifiability which captures the three desired aspects: individual verifiability, universal verifiability and eligibility verifiability. The definition is a sufficient condition for election verifiability, but it may not be

necessary; that is, there could be some protocols which offer verifiability but are not captured by our definition. Our definition is initially presented independently of any particular formal framework. Then, we formalise our definition as reachability assertions in the applied pi calculus. This enables us to realise the first definition of election verifiability which is amenable to automated reasoning, using Blanchet's ProVerif tool [10]. Although election verifiability reduces the set of components required to be trustworthy, in general it does not completely eliminate the need to trust some part of the client software. We therefore introduce the notion of a *sufficiently capable* voting protocol. Intuitively, this is the part of the protocol that needs to "behave correctly" to ensure verifiability. Such parts of the protocol should be implemented using trusted hardware or be auditable. We demonstrate the applicability of our definition by analysing three protocols. The first protocol is a simple illustrative protocol which is trivially verifiable because voters digitally sign their ballot. (Note that this protocol does not achieve other properties such as privacy). We then analyse two protocols from the literature by Fujioka *et al.* [16] and Juels *et al.* [19].

1.2 Related work

Juels, Catalano & Jakobson [18, 19] present the first definition of universal verifiability in the provable security model. Their definition assumes voting protocols produce signature proofs of knowledge demonstrating the correctness of tallying. Automated analysis is not discussed.

Universal verifiability was also studied by Chevallier-Mames et al. [13] with the aim of showing an incompatibility result: protocols satisfying their definition are incompatible with vote-privacy (also called ballot secrecy), and hence coercion-resistance. To see this, note that they require functions f and f' such that for any bulletin board BB and list of eligible voters L the function $f(BB, L)$ returns the list of actual voters and $f'(BB, L)$ returns the election outcome (see Definition 1 of [13]). From these functions one could consider any single bulletin board entry b and compute $f(\{b\}, L)$, $f'(\{b\}, L)$ to reveal a voter and her vote. Our definitions do not reflect such an incompatibility of properties and allow protocols to satisfy both verifiability and coercion-resistance.

Baskar, Ramanujan & Suresh [9] and subsequently Talbi et al. [21] have formalised individual and universal verifiability with respect to the FOO [16] electronic voting protocol. 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).

1.3 Outline

Section 2 recalls the applied pi calculus. In Section 3 we introduce a generic definition of verifiability which is independent of any particular formal framework. In Section 4 our definition is formalised as reachability assertions in the context of the applied pi calculus. In Section 5, we consider what are the minimal trust requirements that are needed to permit election verifiability. The three case studies are analysed in Section 6. Finally, we conclude and give directions for future work (Section 7).

2 Applied pi calculus

The applied pi calculus [2] is a language for modelling concurrent systems and their interactions. It is an extension of the pi calculus which was explicitly designed for modelling cryptographic protocols. For this purpose, the applied pi calculus allows terms to be 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 $a, b, c, k, m, n, s, t, r, \ldots$, 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 F, 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 $x_1, \ldots, x_l$ with $M_1, \ldots, M_l$. The applied pi calculus relies on a simple type system. Terms can be of sort Channel for channel names or Base for the payload sent out on these channels. 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. Extended processes introduce *active*

$P, Q, R ::=$	processes	$A, B, C ::=$	extended processes
0	null process	P	plain process
$P \mid Q$	parallel	$A \mid B$	parallel composition
$!P$	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		
if $M = N$ then P else Q	conditional		

Fig. 1. Applied pi calculus grammar

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.

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. 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.

We introduce *linear* processes as a subset of plain processes generated by the grammar

$$\begin{aligned} P ::= {} & 0 \mid \nu\, n.P \mid c(x).P \mid \overline{c}\langle M\rangle.P \\ & \mid \text{if } M = N \text{ then } P \text{ else } 0 \quad (P \neq 0) \\ & \mid \text{if } M = N \text{ then } 0 \text{ else } P \quad (P \neq 0) \end{aligned}$$

Linear processes can be sequentially composed in a natural way. Let P be a linear processes and Q a plain process. We define the plain process $P \circ Q$ to be Q if $P = 0$ and otherwise by replacing the unique occurrence of ".0" in P by ".Q". (Note that ".0" does not occur in "else 0".) Moreover, we note that if P and Q are both linear processes then $P \circ Q$ is also a linear process.

2.2 Semantics

We now define the operational semantics of the applied pi calculus by the means of two relations: structural equivalence and internal reductions. *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:

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	$\{^M/_x\} \equiv \{^N/_x\}$ if $M =_E N$
PAR-C	$A \mid B \equiv B \mid A$		
NEW-0	$\nu n.0 \equiv 0$	ALIAS	$\nu x.\{^M/_x\} \equiv 0$
NEW-C	$\nu u.\nu w.A \equiv \nu w.\nu u.A$	SUBST	$\{^M/_x\} \mid A \equiv \{^M/_x\} \mid A\{^M/_x\}$

NEW-PAR $\quad A \mid \nu u.B \;\equiv\; \nu u.(A \mid B) \quad$ if $u \notin \mathrm{fn}(A) \cup \mathrm{fv}(A)$

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

COMM	$\overline{c}\langle x\rangle.P \mid c(x).Q \rightarrow P \mid Q$	
THEN	if $M = M$ then P else $Q \rightarrow P$	
ELSE	if $M = N$ then P else $Q \rightarrow Q$	if M, N ground and $M \neq_E N$

2.3 Notational conventions

By convention we assume that the signatures we consider always contains the binary function pair, the unary functions fst, snd and the constant ε. The associated equational theory is defined by $\mathsf{fst}(\mathsf{pair}(x, y)) = x$ and $\mathsf{snd}(\mathsf{pair}(x, y)) = y$. Arbitrary length tuples can be constructed as $\mathsf{pair}(x_1, \mathsf{pair}(x_2, \ldots, \mathsf{pair}(x_n, \varepsilon)))$; which, for convenience, we abbreviate $(x_1, \ldots, x_n)$. We also write M_j for $\mathsf{fst}(\mathsf{snd}(\ldots \mathsf{snd}(M))))$ where $j \geq 1$ and there are $j-1$ occurrences of snd. Finally, we sometimes write $c(x_1, \ldots, x_n)$ (or $c(\tilde{x})$) for the sequence of inputs $c(x_1).\ \ldots\ .c(x_n)$.

To aid readability, we also allow boolean combinations of $M = N$ in conditionals in the "if-then-else" process, and in the output of terms. If ϕ is such a combination, then the output $\overline{c}\langle\phi\rangle.P$ is an abbreviation for "if ϕ then $\overline{c}\langle\mathsf{true}\rangle.P$". Boolean combinations of conditionals in "if" statements are handled as follows. First, the boolean combination is written using only the boolean connectives $\wedge$ and $\neg$. Then $\wedge$ is encoded using nested "if" processes; and negation ($\neg$) is encoded by swapping "then" and "else" branches.

2.4 Events and reachability assertions

For the purpose of protocol analysis processes are annotated with *events* which mark important actions performed by the protocol which do not otherwise affect behaviour. We adopt the formalism presented by Abadi, Blanchet & Fournet [1] to capture events. Events are modelled as outputs $\overline{\mathtt{f}}\langle M\rangle$ where $\mathtt{f} \in \mathcal{F}$ is an *"event channel"*: a name in a particular set $\mathcal{F}$ disjoint from the set of ordinary channels a, b, c. Message input on event channels must use *"event variables"* e, e'.

We assume that protocols should be executed in the presence of a so-called Dolev-Yao adversary [15]. The adversary is permitted to input $\mathtt{f}(e)$ on event channels but is forbidden from using the bound event variable e in any other manner. The former condition prevents processes blocking, whereas the latter ensures the adversary's knowledge cannot be extended by the occurrence of events.

Definition 1 (Adversary). *An* adversary *is a closed process such that, any event channel* $\mathtt{f} \in \mathcal{F}$ *and event variable* e*, only occur in inputs of the form* $\mathtt{f}(e)$.

A reachability assertion is specified as an event $\overline{\mathtt{f}}\langle\tilde{X}\rangle$ where $\tilde{X}$ is a tuple of variables and constants. A process satisfies reachability if there exists an adversary who is able to expose the event (Definition 2). When such an adversary does not exist, we say the process satisfies the unreachability assertion $\overline{\mathtt{f}}\langle\tilde{X}\rangle$.

Definition 2 (Reachability). *The closed process* P *satisfies the* reachability *assertion* $\overline{\mathtt{f}}\langle\tilde{X}\rangle$ *where* $\tilde{X}$ *is a tuple of variables and constants if there exists an adversary* Q *such that* $P \mid Q \rightarrow^* C[\overline{\mathtt{f}}\langle\tilde{X}\rangle.P']$ *for some evaluation context* C *and process* P'.

3 Election verifiability

Election verifiability can be formalised with respect to tests R^{IV}, R^{UV}, R^{EV} corresponding to the three aspects of our formalisation. We assume that the bulletin board output of an election procedure contains a set of ballots $\tilde{T}$, a tuple of votes $\tilde{s}$ representing the election outcome declared result and a set of public voter credentials $\tilde{U}$ where $|\tilde{s}| = |\tilde{T}| = |\tilde{U}|$. The credentials for voters whom abstain are assumed to be omitted from the bulletin board and hence we have the strict equality $|\tilde{T}| = |\tilde{U}|$. Each of the tests R^{IV}, R^{UV}, R^{EV} is a predicate which after substitutions from the bulletin board and elsewhere evaluates to true or false. The designers of electronic voting protocols need not explicitly specify the cryptographic tests since our definition considers the existence of tests (perhaps devised after design) which satisfy our conditions. This extends the applicability of our methodology whilst also permitting the scrutiny of tests specified by protocol designers.

3.1 Overview

Individual verifiability. The test R^{IV} takes parameters v (a vote), $\tilde{x}$ (a voter's knowledge) and z (a bulletin board entry). For R^{IV} to be a suitable test it must allow a voter to identify her bulletin board entry. Formally we require for all votes s, if the voter votes for candidate s, then there exists an execution of the protocol which produces $\tilde{M}$ such that some bulletin board entry T satisfies:

$$R^{IV}\{s/_v, \tilde{M}/_{\tilde{x}}, T/_z\} \tag{1}$$

Moreover, the bulletin board entry should determine the vote; that is, for all bulletin board entries T, votes s, t and tuples $\tilde{M}, \tilde{N}$ we have:

$$R^{IV}\{s/_v, \tilde{M}/_{\tilde{x}}, T/_z\} \wedge R^{IV}\{t/_v, \tilde{N}/_{\tilde{x}}, T/_z\} \Rightarrow (s = t) \tag{2}$$

This ensures the test will only hold for at most one vote.

In addition, individual verifiability requires voters to accept distinct bulletin board entries. This condition requires protocol executions to introduce some freshness (e.g. randomness).

Universal verifiability. This property is encapsulated by the test R^{UV} which takes parameters v (a vote) and z (a bulletin board entry). Given R^{IV}, the test R^{UV} is suitable if every bulletin board entry which is accepted by a voter is also accepted by an observer; and the entry is counted by the observer in the correct way. The property requires that for all executions of the protocol producing $\tilde{M}$ with respect to the voter's vote s, if there exists a bulletin board entry T such that the voter accepts the bulletin board entry as hers, then the observer also accepts the entry:

$$R^{IV}\{s/_v, \tilde{M}/_{\tilde{x}}, T/_z\} \Rightarrow R^{UV}\{s/_v, T/_z\} \tag{3}$$

Moreover, the observer counts the vote correctly. That is for all bulletin board entries T and votes s, t if the test succeeds for s and t then they must be votes for the same candidate:

$$R^{UV}\{s/_v, T/_z\} \wedge R^{UV}\{t/_v, T/_z\} \Rightarrow (s = t) \tag{4}$$

This ensures that an observer may only count a vote in one way.

We remark that the implication in formula 3 is only one way because the adversary is able to construct ballots which would be accepted by an observer. This behaviour can be detected by eligibility verifiability.

Eligibility verifiability. The property is encoded by the test R^{EV} which takes parameters y (a voter's public credential) and z (a bulletin board entry). Given R^{IV}, the test R^{EV} is considered suitable if it ensures: 1) an observer can attribute a bulletin board entry to a public credential if and only if the corresponding voter would accept that entry as hers; and 2) a bulletin board entry can be attributed to at most one voter. Formally the property requires for all bulletin board entries T and executions of the protocol producing $\tilde{M}$ with respect to the voter's vote s and public credential U the voter accepts T as hers iff the bulletin board entry can be attributed to her public credential:

$$R^{IV}\{s/_v, \tilde{M}/_{\tilde{x}}, T/_z\} \Leftrightarrow R^{EV}\{U/_y, T/_z\} \tag{5}$$

This ensures that if R^{IV} succeeds for a voter then she is assured that her vote is considered eligible by an observer; and if R^{EV} succeeds for a public credential then the corresponding voter must have constructed that bulletin board entry. The condition relies upon a relationship between the voter's knowledge $\tilde{M}$ and the voter's public credential U. The second condition requires that the test must uniquely determine who cast a bulletin board entry; that is, for all bulletin board entries T and public voter credentials U, V if the test succeeds for U and V then the credentials are equivalent:

$$R^{EV}\{U/_y, T/_z\} \wedge R^{EV}\{V/_y, T/_z\} \Rightarrow (U = V) \tag{6}$$

This property enables re-vote elimination. The concept of re-voting is particularly useful since it is used by some protocols to provide coercion resistance. In such protocols re-vote elimination is performed with respect to a publicly defined policy to ensure voters vote at most once.

Finally, eligibility verifiability also requires that voters must have unique public credentials and at most one ballot per registered credential may appear in the bulletin board.

The generic definition presented here will be formalised in the context of the applied pi calculus in §4.

3.2 Verifying an election

In addition to proving the verifiability of the electronic voting protocol (§3.1) the voters and observers must be able to check that an arbitrary election was performed in a satisfactory manner. This is achieved by performing the tests R^{IV}, R^{UV} and R^{EV} on the bulletin board. For individual verifiability, each voter should be in possession of her vote t and $\tilde{M}$ representing the knowledge learnt during an execution of the protocol, such that there exists $j \in [1, |\tilde{T}|]$ satisfying the test $R^{IV}\{t/_v, \tilde{M}/_{\tilde{x}}, T_j/_z\}$. For universal verifiability, the bulletin board must be such that the observer can map the ballots to the votes appearing in the election outcome. That is, there exists a bijective function $f : \{1, \ldots, |\tilde{T}|\} \to \{1, \ldots, |\tilde{T}|\}$, such that for all $i \in [1, |\tilde{T}|]$, the test $R^{UV}\{s_i/_v, T_{f(i)}/_z\}$ holds. Similarly, for eligibility verifiability to hold an observer must be able to map each public credential to a bulletin board entry. That is, there exists a bijective function $g : \{1, \ldots, |\tilde{T}|\} \to \{1, \ldots, |\tilde{T}|\}$ such that for all $i \in [1, |\tilde{T}|]$ the test $R^{EV}\{U_i/_y, T_{g(i)}/_z\}$ holds.

Some electronic voting protocols utilise mixnets to obtain privacy. For simplicity we omit formalising the security of mixnets and hence omit modelling the mix. This clearly violates privacy properties. However, since mixnets are verifiable, privacy and election verifiability properties may coexist in practice.

4 Election verifiability in the applied pi calculus

A voting protocol is captured by a voter process P and a process Q modelling administrators whom are required to be honest for the purpose of election verifiability.

Dishonest administrators need not be explicitly modelled since they are part of the adversarial environment. The process Q is assumed to publish voter credentials. Channels $\tilde{a}$ are assumed to be private and appear in both P, Q. In addition, we consider a context C which performs setup duties; for example, the instantiation of keys for honest administrators. Dishonest administrator keys are modelled as free names. Definition 3 formalises a voting process specification accordingly. The definition allows us to analyse election verifiability with respect to an unbounded number of voters and arbitrarily many candidates.

Definition 3 (Voting process specification). *A* voting process specification *is a tuple* $\langle C, P, Q[\overline{c}\langle U\rangle], \tilde{a}\rangle$ *where* P *is a linear process,* C *and* Q *are contexts such that* P, C, Q *do not contain any occurrence of event channels and event variables.* U *is a term modelling public voter credential. The variable* $v \in fv(P)$ *refers to the value of the vote,* $v \notin bv(C) \cup bv(P)$*,* $c \notin (\tilde{a} \cup bn(C[Q]))$*,* $(fv(P)\backslash\{v\} \cup fv(Q)) \subseteq bv(C)$ *and* $\{x_n \mid n \in bn(P)\} \cap bv(P) = \emptyset$*.*

Given a voting process specification $\langle C, P, Q[\overline{c}\langle U\rangle], \tilde{a}\rangle$ we can "put the pieces together" and obtain the process modelling the voting protocol

$$\mathsf{VP} \mathrel{\widehat{=}} \nu\, b.(C[!\nu\, \tilde{a}.(b(v).P \mid Q[\overline{c}\langle U\rangle])] \mid !\nu\, s.((!\overline{b}\langle s\rangle) \mid \overline{c}\langle s\rangle))$$

where $b \notin (\tilde{a} \cup \mathrm{fn}(C[P \mid Q]) \cup \mathrm{bn}(C[P \mid Q]))$. Intuitively, the channel b is used to communicate to each voter the instantiation of his vote v. The process $!\nu\, s.((!\overline{b}\langle s\rangle) \mid \overline{c}\langle s\rangle)$ models the generation of all possible choices of votes. The nested replication allows several voters to use the same s while other voters use different values. Each vote s is also made available to the environment by publishing it on the channel c.

The formalisation of election verifiability (Definition 5) can naturally be expressed as reachability assertions [23, 10] associated with the propositional formulas 1-6 of §3.1 which relate to tests R^{IV}, R^{UV}, R^{EV}. First the tests must be incorporated into an augmented voting process (Definition 4).

Definition 4 (Augmented voting process). *Given a voting process specification* $\langle C, P, Q[\overline{c}\langle U\rangle], \tilde{a}\rangle$ *and tests* R^{IV}, R^{UV}, R^{EV} *the* augmented voting process *is defined as* $\mathcal{P} = \nu\, b.(C[!\nu\, \tilde{a}, b'.(\hat{P} \mid \hat{Q})] \mid R \mid R') \mid R'' \mid R'''$ *where*

$$\begin{aligned}
\hat{P} &= b(v).P \circ c(z).b'(y).(\overline{\mathtt{pass}}\langle (R^{IV}\tau, z)\rangle \mid \overline{\mathtt{fail}}\langle\psi\rangle)\\
\hat{Q} &= Q[\overline{b'}\langle U\rangle \mid \overline{\mathtt{cred}}\langle U\rangle \mid \overline{c}\langle U\rangle]\\
R &= !\nu\, s.((!\overline{b}\langle s\rangle) \mid \overline{c}\langle s\rangle)\\
R' &= b(v').b(v'').c(\tilde{x}').c(\tilde{x}'').c(y').c(y'').c(z').\overline{\mathtt{fail}}\langle\phi' \vee \phi'' \vee \phi'''\rangle\\
R'' &= \mathtt{pass}(e).\mathtt{pass}(e').\overline{\mathtt{fail}}\langle e_1 \wedge e_1' \wedge (e_2 = e_2')\rangle\\
R''' &= \mathtt{cred}(e).\mathtt{cred}(e').\overline{\mathtt{fail}}\langle e = e'\rangle
\end{aligned}$$

$$\begin{aligned}
\psi &= (R^{IV} \wedge \neg R^{UV}) \vee (R^{IV} \wedge \neg R^{EV}) \vee (\neg R^{IV} \wedge R^{EV})\\
\phi' &= R^{IV}\{{}^{v'}/_{v}, {}^{\tilde{x}'}/_{\tilde{x}}, {}^{z'}/_{z}\} \wedge R^{IV}\{{}^{v''}/_{v}, {}^{\tilde{x}''}/_{\tilde{x}}, {}^{z'}/_{z}\} \wedge \neg(v' = v'')\\
\phi'' &= R^{UV}\{{}^{v'}/_{v}, {}^{z'}/_{z}\} \wedge R^{UV}\{{}^{v''}/_{v}, {}^{z'}/_{z}\} \wedge \neg(v' = v'')\\
\phi''' &= R^{EV}\{{}^{y'}/_{y}, {}^{z'}/_{z}\} \wedge R^{EV}\{{}^{y''}/_{y}, {}^{z'}/_{z}\} \wedge \neg(y' =_E y'')
\end{aligned}$$

such that $\mathtt{fail}, \mathtt{pass}, \mathtt{cred}$ *are event channels,* $\tau = \{^n/_{x_n} \mid n \in bn(P) \wedge x_n \in fv(R^{IV})\}$, $\tilde{x} = (bv(P) \cap (fv(Riv) \backslash \{z\})) \cup \{x_n \mid n \in bn(P) \wedge x_n \in fv(Riv)\}$, $b, b' \notin (\tilde{a} \cup fn(C[P \mid Q]) \cup bn(C[P \mid Q]))$ *and* $c \notin (\tilde{a} \cup bn(C[P]))$.

The augmented voting process extends P to bind the voter's intended vote v, assigns the voter's public credential to y and introduces a claimed bulletin board entry z. As in the non-augmented process, the process R produces candidates for whom the voters are allowed to vote. The number of candidates and for whom each voter casts her vote is controlled by the adversarial environment. The events capture the desired reachability assertions. That is, reachability of $\overline{\mathtt{pass}}\langle(R^{IV}, z)\rangle$ captures propositional formula 1 of §3.1; unreachability of $\overline{\mathtt{fail}}\langle\psi\rangle$ models propositional formulas 3 & 5 of §3.1; and unreachability of $\overline{\mathtt{fail}}\langle\phi' \vee \phi'' \vee \phi'''\rangle$ denotes formulas 2, 4 & 6 of §3.1. The universal quantifiers of the propositional formulas 2-6 are captured by allowing the adversary to input the required parameters. Process R'' exploits communication on event channels to detect the scenario in which two voters accept the same bulletin board entry. Similarly, communication on the event channel $\mathtt{cred}$ is used to detect the situation in which two voters are assigned the same public credential.

Definition 5 (Election verifiability). *A voting process specification* $\langle C, P, Q[\overline{c}\langle U\rangle], \tilde{a}\rangle$ *satisfies* election verifiability *if there exists tests* R^{IV}, R^{UV}, R^{EV} *such that the augmented voting process* $\mathcal{P}$ *and tests satisfy the following conditions:*

1. $\mathcal{P}$ *satisfies the unreachability assertion:* $\overline{\mathtt{fail}}\langle\mathsf{true}\rangle$.
2. $\mathcal{P}$ *satisfies the reachability assertion:* $\overline{\mathtt{pass}}\langle(\mathsf{true}, x)\rangle$.
3. *The tests* R^{IV}, R^{UV}, R^{EV} *satisfy the following constraints:*
   - $fv(R^{IV}) \subseteq bv(P) \cup \{v, z\} \cup \{x_n \mid n \in bn(P)\}$
   - $fv(R^{UV}) \subseteq \{v, z\}$
   - $fv(R^{EV}) \subseteq \{y, z\}$
   - $(fn(R^{IV}) \cup fn(R^{UV}) \cup fn(R^{EV})) \cap bn(\mathcal{P}) = \emptyset$

The reachability assertion $\overline{\mathtt{pass}}\langle(\mathsf{true}, x)\rangle$ represents the voter's ability to identify her bulletin board entry with respect to her vote.

Many protocols in literature do not provide eligibility verifiability. We therefore define a *weakly augmented voting process* and *weak election verifiability* to capture only individual and universal verifiability. A *weakly augmented voting process* is defined as an augmented voting process but with $R''' = 0$, $\psi = (R^{IV} \wedge R^{UV})$ and replacing $\overline{\mathtt{fail}}\langle\phi' \vee \phi'' \vee \phi'''\rangle$ with $\overline{\mathtt{fail}}\langle\phi' \vee \phi''\rangle$. The definition of *weak election verifiability* is obtained by omitting conditions on R^{EV} from Definition 5.

5 Voting on Satan's computer

In *Programming Satan's Computer*, Anderson & Needham [5] describe the task of designing security protocols as programming *"a computer which gives answers which are subtly and maliciously wrong at the most inconvenient possible moment."* In the context of voting systems one should indeed not trust the voting software and the terminal executing it. A virus might for instance change the vote entered by the user or even execute a completely different protocol than the one expected. Ideally, verifiability should

enable voters not to put any trust in the software. While it seems difficult to design protocols where this kind of absolute verifiability is achieved, in practice one may accept to trust some parts of the protocol that rely on trusted hardware or that can be audited. For example in the Helios voting protocol [3], the construction of the ballots can be audited using a cast-or-audit mechanism. Any third party software can be used to audit the ballots and the voter is ensured that the cast ballots were constructed according to the protocol specification with high probability.

Our aim is to identify the parts of the protocol that msut be trusted to achieve verifiability. In the previous section, administrators that did not need to be trusted for verifiability were not specified and assimilated to the environment. We now go a step further and allow parts of the voting protocol to be executed by the environment. Ideally, one would just model the inputs and outputs the voter enters and receives from the voting terminal. The trusted computation components can be encoded as a voting process specification while all the remaining parts will be controlled by the adversarial environment. We say that such a specification is *sufficiently capable* with respect to the original protocol if the same tests are suitable for both.

Definition 6 (Sufficiently capable voting protocol specification). *Let* $\langle C, P, Q[\overline{c}\langle U\rangle], \tilde{a}\rangle$ *be a voting protocol specification which satisfies election verifiability with respect to tests* R^{IV}, R^{UV}, R^{EV}. *We say that* $\langle \bar{C}, \bar{P}, \bar{Q}[\overline{c}\langle U\rangle], \tilde{a}'\rangle$ *is a* sufficiently capable voting protocol specification *if* $\langle \bar{C}, \bar{P}, \bar{Q}[\overline{c}\langle U\rangle], \tilde{a}'\rangle$ *satisfies election verifiability with respect to the same tests* R^{IV}, R^{UV}, R^{EV}.

6 Case studies

We demonstrate the applicability of our methodology by analysing electronic voting protocols from literature. The ProVerif tool [10] has been used for automation and our input scripts are available online[4]. ProVerif's ability to reason with reachability assertions is sound (when no trace is found the protocol is guaranteed to satisfy the unreachability assertion) but not complete (false reachability traces may be found). As a consequence reachability traces output by ProVerif for Condition 2 of Definition 5 must be checked by hand. In this paper all such traces correspond to valid reachable states.

6.1 Postal ballot protocol

Description. Consider an electronic variant of a *"postal ballot"* (or *"mail-in ballot"*) protocol whereby a voter receives her private signing key sk_V from a keying authority, constructs her signed ballot and sends it to the bulletin board. The keying authority is also responsible for publishing the voter's public verification key $\mathsf{pk}(sk_V)$ which will serve as her public credential. The protocol does not satisfy all of the desirable electronic voting properties; but it should certainly provide election verifiability.

[4] `http://www.bensmyth.com/publications/09wissec/`

Formalisation in applied pi. The corresponding voting process specification is given by $\langle C, P, Q[\overline{c}\langle U\rangle], (a)\rangle$ where

$$\begin{array}{ll} C \mathrel{\widehat{=}} \_ & Q[\overline{c}\langle U\rangle] \mathrel{\widehat{=}} \nu\, sk_V.(\overline{a}\langle sk_V\rangle \mid \overline{c}\langle U\rangle) \\ P \mathrel{\widehat{=}} a(x).\overline{c}\langle(v, \mathsf{sign}(v, x))\rangle & U \mathrel{\widehat{=}} \mathsf{pk}(sk_V) \end{array}$$

We model digital signatures (without message recovery) by the equation

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

The resulting (non-augmented) voting process is then defined as

$$\begin{array}{rl} \mathsf{VP}_{\mathsf{postal}} \mathrel{\widehat{=}} \nu b.\big(& !\nu a.\ (b(v).a(x).\overline{c}\langle(v, \mathsf{sign}(v, x))\rangle \\ & \quad \mid \nu\, sk_V.(\overline{a}\langle sk_V\rangle \mid \overline{c}\langle\mathsf{pk}(sk_V)\rangle)) \\ & \mid !\nu s.(!\overline{b}\langle s\rangle \mid \overline{c}\langle s\rangle)\big) \end{array}$$

Analysis. The augmented voting process $\mathcal{P}$ can be derived with respect to tests:

$$\begin{array}{l} R^{IV} \mathrel{\widehat{=}} z =_E (sign(v, x), v, pk(x)) \\ R^{UV} \mathrel{\widehat{=}} \mathsf{checksign}(z_1, z_2, z_3) =_E \mathsf{true} \wedge v =_E z_2 \\ R^{EV} \mathrel{\widehat{=}} \mathsf{checksign}(z_1, z_2, z_3) =_E \mathsf{true} \wedge y =_E z_3 \end{array}$$

ProVerif is able to automatically verify the protocol satisfies election verifiability.

6.2 Protocol due to Fujioka, Okamoto & Ohta

Description. The FOO protocol [16] involves voters, a registrar and a tallier. The protocol relies on a commitment scheme and blind signatures which we model by the following equational theory.

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

The voter first computes her ballot as a commitment to her vote $m' = \mathsf{commit}(v, r)$ and sends the signed blinded ballot $\mathsf{sign}(\mathsf{blind}(m', r'), sk_V)$ to the registrar. The registrar checks the signature belongs to an eligible voter and returns $\mathsf{sign}(\mathsf{blind}(m', r'), sk_R)$ the blind signed ballot. The voter verifies that this input (variable bsb in the process P below) corresponds to the registrar's signature and unblinds the message to recover her ballot signed by the registrar $m = \mathsf{sign}(m', sk_R)$. 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 record[5]. The voter then checks the bulletin board for her entry, the triple (l, m', m), (modelled in P below by the input in variable bbe) and appends the commitment factor r. Finally, using r the tallier opens all of the ballots and announces the declared outcome. The protocol claims to provide individual and universal verifiability but does not consider eligibility verifiability.

[5] The value l is used for practical purposes only; it does not affect security.

Formalisation in applied pi. The voting process specification is $\langle \_, P, Q[\overline{c}\langle \mathsf{pk}(x)\rangle], (a)\rangle$ where $Q = c(x).(\overline{a}\langle x\rangle \mid \_)$ and P is defined as follows:

$$
\begin{array}{l}
P = \nu\, r, r'.\\
\quad a(x).\\
\quad \text{let } m' = \mathsf{commit}(v, r) \text{ in}\\
\quad \overline{c}\langle(\mathsf{pk}(x), \mathsf{blind}(m', r'), \mathsf{sign}(\mathsf{blind}(m', r'), x))\rangle.\\
\quad c(bsb).\\
\quad \text{if } \mathsf{checksign}(bsb, \mathsf{blind}(m', r'), \mathsf{pk}(sk_R)) = \mathsf{true} \text{ then}\\
\quad \text{let } m = \mathsf{unblind}(bsb, r') \text{ in}\\
\quad \overline{c}\langle(m', m)\rangle.\\
\quad c(bbe).\\
\quad \text{if } m' = bbe_2 \wedge m = bbe_3 \text{ then}\\
\quad \overline{c}\langle(bbe_1, r)\rangle
\end{array}
$$

Analysis. Let tests R^{IV}, R^{UV} be defined as follows:

$$
\begin{array}{l}
R^{IV} \mathrel{\widehat{=}} z =_E \langle bbe_1, \mathsf{commit}(v, x_r), \mathsf{unblind}(bsb, x_{r'}), x_r, v\rangle\\
\qquad \wedge\, \mathsf{checksign}(z_3, z_2, \mathsf{pk}(sk_R)) =_E \mathsf{true}\\
R^{UV} \mathrel{\widehat{=}} z_2 =_E \mathsf{commit}(z_5, z_4) \wedge \mathsf{checksign}(z_3, z_2, \mathsf{pk}(sk_R)) =_E \mathsf{true} \wedge z_5 =_E v
\end{array}
$$

ProVerif enables automatic verification of election verifiability with respect weak election verifiability.

Sufficiently capable voting protocol specification. We consider the voting process specification $\langle \_, \nu\, r.\overline{c}\langle r\rangle.c(x_{r'}, bsb, bbe_1), Q[\mathsf{pk}(x)], ()\rangle$. The process $\nu\, r.\overline{c}\langle r\rangle.c(x'_{r'}, bsb, bbe_1)$ corresponds to the trusted construction of nonce r before providing it (outputting) to the untrusted voting terminal which returns (inputs) the resulting computation. Note that the input $c(x_{r'}, bsb, bbe_1)$ ensures that the test R^{IV} is closed inside the augmented voting process. ProVerif is able to verify that the specification is sufficiently capable. It follows that election verifiability can be achieved by a voter who is able to construct the fresh commitment factor r, while no other computation needs to be trusted for verifiability. Other properties, for example anonymity, may of course require different parts of the protocol to be trusted.

6.3 Protocol due to Juels, Catalano & Jakobsson and Clarkson, Chong & Myers

Description. The protocol due to Juels, Catalano & Jakobsson [19], which has been implemented by Clarkson, Chong & Myers [**?**,**?**], involves voters, registrars and talliers. The registrars provide each voter with a credential k and publishes the encrypted credential $\mathsf{penc}(k, r'', \mathsf{pk}(sk_T))$. The registrars are also responsible for announcing the candidate list $\tilde{s} = (s_1, \ldots, s_l)$. To cast her ballot the voter selects her vote $s \in \tilde{s}$ and computes the ElGamal ciphertexts $M = \mathsf{penc}(s, r, \mathsf{pk}(sk_T))$, $M' = \mathsf{penc}(k, r', \mathsf{pk}(sk_T))$. The first ciphertext contains her vote and the second contains her credential. In addition, the voter constructs a signature proof of knowledge N demonstrating the correct construction of her ciphertexts and that she has chosen a valid candidate; that is, $s \in \tilde{s}$. The voter posts her ciphertexts and signature proof of knowledge to the bulletin board.

After some predefined deadline the outcome is computed. The talliers begin by discarding any entries for which signature proofs of knowledge do not hold and eliminate re-votes by performing pairwise plaintext equality tests[6] on all the ciphertexts containing voting credentials. A verifiable mix is then performed on the bulletin board. Entries based on invalid voter credentials are then weeded using plaintext equality tests between the entries posted to the bulletin board by the mix and those published by the registrar. Re-vote elimination is performed in a verifiable manner with respect to some publicly defined policy and ballot weeding is achieved similarly. The verifiable nature of these steps is essential. Finally, the talliers perform a verifiable decryption and publish the result.

Formalisation in applied pi. The formalism for signature proofs of knowledge due to Backes *et al.* [7, 8] will be adopted. A signature proof of knowledge is a term $\mathsf{spk}_{i,j}(\tilde{U}, \tilde{V}, F)$ where $\tilde{U} = (U_1, \ldots, U_i)$ denotes the witness (or private component), $\tilde{V} = (V_1, \ldots, V_j)$ defines the public parameters and F is a formula over those terms. More precisely F is a term without names or variables, but includes distinguished constants α_k, β_l where $k, l \in \mathbb{N}$. The constants α_k, β_l in F denote placeholders for the terms $U_k \in \tilde{U}$, $V_l \in \tilde{V}$ used within a signature of knowledge $\mathsf{spk}_{i,j}(\tilde{U}, \tilde{V}, F)$. For example, the signature proof of knowledge used by voters in the Juels, Catalano & Jakobsson voting protocol [19] demonstrates possession of a vote s, credential k and randomisation factors r, r' such that $M = \mathsf{penc}(s, r, \mathsf{pk}(sk_T))$, $M' = \mathsf{penc}(k, r', \mathsf{pk}(sk_T))$ and $s \in \tilde{s}$; that is, the proof shows the ciphertexts are correctly formed and s is a valid candidate. This can be captured by $\mathsf{spk}_{4,3+l}((s, r, k, r'), (M, M', \mathsf{pk}(sk_T), s_1, \ldots, s_l), \mathcal{F})$ where $\mathcal{F}$ is defined as $\beta_1 = \mathsf{penc}(\alpha_1, \alpha_2, \beta_3) \wedge \beta_2 = \mathsf{penc}(\alpha_3, \alpha_4, \beta_3) \wedge (\alpha_1 = \beta_4 \vee \ldots \vee \alpha_1 = \beta_{4+l})$. A term $\mathsf{spk}_{i,j}(\tilde{U}, \tilde{V}, F)$ represents a valid signature if the term obtained by substituting U_k, V_l for the corresponding α_k, β_l evaluates to true. Verification of such a statement is modelled by the function $\mathsf{ver}_{i,j}$. The equational theory includes the following equations over all $i, j \in \mathbb{N}$, tuples $\tilde{x} = (x_1, \ldots, x_i), \tilde{y} = (y_1, \ldots, y_j)$ and formula F which is a ground term over $\Sigma \cup \{\alpha_k, \beta_l \mid k \leq i,\, l \leq j\}$ without any names:

$$\begin{aligned} \mathsf{public_p}(\mathsf{spk}_{i,j}(\tilde{x}, \tilde{y}, F)) &= y_p \text{ where } p \in [1, j] \\ \mathsf{formula}(\mathsf{spk}_{i,j}(\tilde{x}, \tilde{y}, F)) &= F \end{aligned}$$

In addition, we define equations such that $\mathsf{ver}_{i,j}(F, \mathsf{spk}_{i,j}(\tilde{U}, \tilde{V}, F')) =_E \mathsf{true}$ if $F =_E F'$ and $F\{{}^{U_1}/_{\alpha_1}, \ldots, {}^{U_i}/_{\alpha_i}, {}^{V_1}/_{\beta_1}, \ldots, {}^{V_j}/_{\beta_j}\}$ holds where $i = |\tilde{U}|$, $j = |\tilde{V}|$ and F, F' are ground terms over $\Sigma \cup \{\alpha_k, \beta_l \mid k \leq i,\, l \leq j\}$ without names. We omit the details of these equations which are similar to [7, 8] due to lack of space.

The protocol makes use of the ElGamal encryption scheme. Decryption and re-encryption are captured accordingly:

$$\begin{aligned} \mathsf{dec}(\mathsf{penc}(x, y, \mathsf{pk}(z)), z) &= x \\ \mathsf{renc}(\mathsf{penc}(x, y, z), y') &= \mathsf{penc}(x, f(y, y'), z) \end{aligned}$$

The ElGamal scheme also exhibits the feature expressed by the equation:

$$\mathsf{dec}(\mathsf{penc}(x, y, \mathsf{pk}(z)), \mathsf{commit}(\mathsf{penc}(x, y, \mathsf{pk}(z)), z)) = x$$

[6] A *plaintext equality test* [17] is a cryptographic predicate which allows the comparison of two ciphertexts. The test returns true if the ciphertexts contain the same plaintext.

Verifiable decryption can then be achieved using the signature proof of knowledge $\mathsf{spk}_{1,3}((\alpha_1),(\beta_1,\beta_2),\mathcal{F}')$ where $\mathcal{F}'$ is given by $\beta_1 = \mathsf{commit}(\beta_2,\alpha_1)$. The proof shows that if $\beta_2 = \mathsf{penc}(M,N,\mathsf{pk}\alpha_1)$ for some terms M,N, then β_1 is a decryption key for β_2. Finally, plaintext equality tests are modelled by the equation

$$\mathsf{pet}(\mathsf{penc}(x,y,pk(z)),\mathsf{penc}(x,y',\mathsf{pk}(z)),\\ \mathsf{petkey}(\mathsf{penc}(x,y,\mathsf{pk}(z)),\mathsf{penc}(x,y',\mathsf{pk}(z)),z)) = \mathsf{true}$$

The voting process specification can now be defined as $\langle C,P,Q[\overline{c}\langle U\rangle],(a)\rangle$ where $U = \mathsf{penc}(k,r'',\mathsf{pk}(sk_T))$ and C,P,Q are specified as follows:

$$\begin{array}{ll}
C & = \nu\, sk_T.(\overline{c}\langle\mathsf{pk}(sk_T)\rangle \mid (!Q') \mid (!Q'') \mid _) \\
P & = \nu\, r.\nu\, r'.a(x). \\
 & \quad \text{let } M = \mathsf{penc}(v,r,\mathsf{pk}(sk_T)) \text{ in} \\
 & \quad \text{let } M' = \mathsf{penc}(x,r',\mathsf{pk}(sk_T)) \text{ in} \\
 & \quad \text{let } N = \mathsf{spk}_{4,3+l}((v,r,x,r'),(M,M',\mathsf{pk}(sk_T),s_1,\ldots,s_l),\mathcal{F}) \text{ in} \\
 & \quad \overline{c}\langle(M,M',N)\rangle \\
Q & = \nu\, k.\nu\, r''.(\overline{a}\langle k\rangle \mid _) \\
Q' & = c(y).\overline{c}\langle\mathsf{petkey}(y_1,y_2,sk_T)\rangle \\
Q'' & = c(z).\text{if } \mathsf{ver}_{4,3+l}(\mathcal{F},z) = \mathsf{true} \text{ then} \\
 & \quad \overline{c}\langle\mathsf{spk}_{1,2}((sk_T),(\mathsf{commit}(\mathsf{public}_1(z),sk_T),\mathsf{public}_1(z)),\mathcal{F}')\rangle)
\end{array}$$

For simplicity we consider a single registrar Q and tallier $(Q' \mid Q'')$ whom are assumed to be honest. Moreover, we assume the existence of a secure mix protocol and hence do not model or verify the shuffle. The registrar process is standard and is responsible for publishing the public voter credentials. Process Q' captures the tallier's responsibility to provide suitable keys for plaintext equality tests used for ballot weeding and re-vote elimination. Finally, process Q'' encodes the tallier's willingness to provide verifiable decryption for honestly constructed ballots.

Analysis. The protocol is dependent on the candidate list and therefore cannot be verified with respect to an arbitrary number of candidates. We must therefore restrict the adversarial environment to consider a fixed number of candidates. This can be achieved by modifying the construction of an augmented process; more precisely we redefine R as $(!\overline{b}\langle s_1\rangle) \mid \ldots \mid (!\overline{b}\langle s_l\rangle)$ where $s_1,\ldots,s_l$ are free names representing the candidates for whom voters may cast their votes. Let the tests R^{IV},R^{UV},R^{EV} be defined as follows:

$$\begin{array}{ll}
R^{IV} & \widehat{=}\ \phi' \wedge z_1 =_E \mathsf{spk}_{4,3+l}((v,x_r,x_k,x_{r'}),(M,M',\mathsf{pk}(sk_T),s_1,\ldots,s_l),\mathcal{F}) \\
R^{UV} & \widehat{=}\ \phi\ \wedge \mathsf{dec}(\mathsf{public}_2(z_2),\mathsf{public}_1(z_2)) =_E v \\
R^{EV} & \widehat{=}\ \phi' \wedge \mathsf{ver}_{4,3+l}(\mathcal{F},z_1) =_E \mathsf{true}
\end{array}$$

$$\begin{array}{ll}
\phi & \widehat{=}\ \mathsf{ver}_{1,2}(\mathcal{F}',z_2) =_E \mathsf{true} \wedge \mathsf{public}_1(z_1) =_E \mathsf{public}_2(z_2) \\
\phi' & \widehat{=}\ \phi \wedge \mathsf{pet}(y,\mathsf{public}_2(z_1),z_3) =_E \mathsf{true}
\end{array}$$

where $M = \mathsf{penc}(v,x_r,\mathsf{pk}(sk_T))$ and $M' = \mathsf{penc}(x_k,x_{r'},\mathsf{pk}(sk_T))$. The augmented voting process can now be derived with respect to the size of the candidate list l. Using

ProVerif in association with a PHP script that generates ProVerif scripts for different values of l, the protocol can be successfully verified to satisfy election verifiability with respect to $l \in [1, 100]$.

Sufficiently capable voting process specification. Consider the voting process specification $\langle \_, \bar{P}, \bar{Q}[\overline{c}\langle y\rangle], (a)\rangle$ where $\bar{P} = a(x).c(x_r, x_{r'})$ and $\bar{Q} = \nu\, k.\overline{c}\langle k\rangle.c(y).(\overline{a}\langle k\rangle \mid \_)$. The specification defines a registrar who constructs a credential k and sends it to the voter on a private channel. The voter then inputs two (supposed) randomisation factors from the untrusted voting terminal. In addition, the registrar inputs the voter's public credential from untrusted computation and publishes it. The individual and universal aspects of election verifiability have been successfully verified using ProVerif. For eligibility verifiability consider the voting process specification $\langle C, \bar{P}', \bar{Q}'[\overline{c}\langle y\rangle], (a)\rangle$ where:

$$\bar{P}' = a(x).c(x_r, x_{r'}).\overline{c}\langle \mathsf{spk}_{4,3+l}((v, x_r, x, x_{r'}), (M, M', \mathsf{pk}(sk_T), s_1, \ldots, s_l), \mathcal{F})\rangle$$
$$\bar{Q}' = \nu\, k.\overline{c}\langle k\rangle.c(x_{r''}).\text{let } y = \mathsf{penc}(k, x_{r''}, \mathsf{pk}(sk_T)) \text{ in } (\overline{a}\langle k\rangle \mid \_)$$

which is sufficiently capable with respect to all three aspects of our definition. It follows that election verifiability can be achieved in the presence of a public key infrastructure; an honest registrar who is able to generate a random private credential k and perform a single ElGamal encryption; and finally, the voter is able to compute signatures of knowledge and construct a pair of ElGamal encryptions.

6.4 Discussion of trust requirements

The notion of voting using "Satan's computer" allows the relative degrees of election verifiability to be assessed. In this paper we studied a postal ballot protocol (as defined in §6.1), the FOO protocol [16] and the protocol due to Juels, Catalano & Jakobsson [19]. Our analysis has found that no assumptions need be made to achieve universal verifiability. In the postal ballot protocol a public key infrastructure (PKI) is required for individual verifiability whereas only secure random generation is required for [16] and [19]. More precisely, [16] is dependent on the voter's ability to generate the random commitment factor r and [19] requires the registrar to construct a random private credential k. Eligibility verifiability is provided by the postal ballot protocol and [19], both of which require a public key infrastructure. In addition, [19] requires an honest administrator whom can generate secure randoms and perform ElGamal encryption; and the voter is able to construct ElGamal encryptions and compute signatures of knowledge.

7 Conclusion

This paper presents a formal definition of election verifiability for electronic voting protocols. The idea of tests for individual, universal and eligibility verifiability (and the associated acceptability conditions) is independent of any particular formalism. We instantiate this idea in terms of reachability assertions in the context of the applied pi calculus. The definition is suitable for automated reasoning using the ProVerif software tool. The applicability of our work has been demonstrated by the analysis of protocols

from literature; namely Fujioka, Okamoto & Ohta [16] and Juels, Catalano & Jakobsson [19] which has been implemented by Clarkson, Chong & Myers [**?**,**?**]. Since the latter protocol also satisfies vote-privacy, receipt-freeness and coercion resistance [19, 6] our definition is compatible with these properties, in contrast with the definition of [13]. We also discuss sufficient voter capabilities for these protocols to satisfy verifiability.

In future work, we intend to generalise our definitions of universal verifiability and eligibility verifiability to work with voting systems that rely on homomorphic combination of encrypted votes (such as Helios 2.0 [4]). We also aim to formalise the security of weeding policies for re-voting, present in many realistic systems, which we omitted from our case studies.

Acknowledgements

We are particularly grateful to Michael Clarkson for careful reading of an earlier draft, and for his perceptive questions and comments. In addition, we would like to thank the anonymous WISSec'09 reviewers for helpful remarks.

References

1. M. Abadi, B. Blanchet, and C. Fournet. Just fast keying in the pi calculus. *ACM Transactions on Information and System Security (TISSEC)*, 10(3):1–59, July 2007.
2. 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.
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: Proceedings of the 21st IEEE Computer Security Foundations Symposium*, pages 195–209, Washington, USA, 2008. IEEE.
7. M. Backes, M. Maffei, and D. Unruh. Zero-Knowledge in the Applied Pi-calculus and Automated Verification of the Direct Anonymous Attestation Protocol. Cryptology ePrint Archive: Report 2007/289, July 2007.
8. M. Backes, M. Maffei, and D. Unruh. Zero-Knowledge in the Applied Pi-calculus and Automated Verification of the Direct Anonymous Attestation Protocol. In *S&P'08: Proceedings of the 2008 IEEE Symposium on Security and Privacy*, pages 202–215, Washington, DC, USA, 2008. IEEE Computer Society.
9. 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.
10. B. Blanchet. Automatic verification of correspondences for security protocols. *Journal of Computer Security*, 2009. To appear.

11. 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.
12. 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.
13. 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.
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. D. Dolev and A. C. Yao. On the security of public key protocols. *Information Theory*, 29:198–208, 1983.
16. 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.
17. 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.
18. A. Juels, D. Catalano, and M. Jakobsson. Coercion-Resistant Electronic Elections. Cryptology ePrint Archive, Report 2002/165, 2002.
19. 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.
20. 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.
21. 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.
22. UK Electoral Commission. Key issues and conclusions: May 2007 electoral pilot schemes. http://www.electoralcommission.org.uk/elections/pilots/May2007.
23. T. Y. C. Woo and S. S. Lam. A semantic model for authentication protocols. In *S&P'93: Proceedings of the 1993 IEEE Symposium on Security and Privacy*, pages 178–194, Washington, DC, USA, 1993. IEEE Computer Society.