Election verifiability in electronic voting protocols* **

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

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

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

1 Introduction

Electronic voting systems are being introduced, or trialled, in several countries to provide more efficient voting procedures. However, the security of electronic elections has been seriously questioned [9, 20, 8, 24]. A major difference with traditional paper based elections is the lack of transparency. In paper elections it is often possible to observe the whole process from ballot casting to tallying, and to rely on robustness characteristics of the physical world (such as the impossibility of altering the markings on a paper ballot sealed inside a locked ballot box). By comparison, it is not possible to observe the electronic operations performed on data. Computer systems may alter voting records in a way that cannot be detected by either voters or election observers. A voting terminal's software might be infected by malware which could change the entered vote, or even execute a completely different protocol than the one expected. The situation can be described as *voting on Satan's computer*, analogously with [5].

The concept of *election* or *end-to-end verifiability* that has emerged in the academic literature, *e.g.*, [17, 18, 10, 3, 21, 2], aims to address this problem. It should allow voters and election observers to verify, independently of the hardware and software running the election, that votes have been recorded, tallied and declared correctly. One generally distinguishes two aspects of verifiability.

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

** A long version containing full proofs is available in [19].

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

We identify another aspect that is sometimes included in universal verifiability.

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

We explicitly distinguish eligibility verifiability as a distinct property.

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

The consideration of eligibility verifiability is particularly interesting as it provides an assurance that the election outcome corresponds to votes legitimately cast and hence provides a mechanism to detect ballot stuffing. We note that this property has been largely neglected in previous work and our earlier work [22] only provided limited scope for.

A further interesting aspect of our work is the clear identification of which parts of the voting system need to be trusted to achieve verifiability. As it is not reasonable to assume voting systems behave correctly we only model the parts of the protocol that we need to trust for the purpose of verifiability; all the remaining parts of the system will be controlled by the adversarial environment. Ideally, such a process would only model the interaction between a *voter* and the voting terminal; *that is, the messages input by the voter*. In particular, the voter should not need to trust the election hardware or software. However, achieving absolute verifiability in this context is difficult and protocols often need to trust some parts of the voting software or some administrators. Such trust assumptions are motivated by the fact that parts of a protocol can be audited, or can be executed in a distributed manner amongst several different election officials. For instance, in Helios 2.0 [3], the ballot construction can be audited using a cast-or-audit mechanism. Whether trust assumptions are reasonable depends on the context of the given election, but our work makes them explicit.

Tests Φ^{IV}, Φ^{UV} and Φ^{EV} are assumed to be verified in a trusted environment (if a test is checked by malicious software that always evaluates the test to hold, it is useless). However, the verification of these tests, unlike the election, can be repeated on different machines, using different software, provided by different stakeholders of the election. Another possibility to avoid this issue would be to have tests which are human-verifiable as discussed in [2, Chapter 5].

We apply our definition on three case studies: the protocol by Fujioka *et al.* [15]; the Helios 2.0 protocol [4] which was effectively used in recent university elections in Belgium; and the protocol by Juels *et al.* [18], which has been implemented by Clarkson *et al.* as Civitas [13, 12]. This demonstrates that our definition is suitable for a large class of protocols; including schemes based on mixnets, homomorphic encryption and blind signatures. (In contrast, our preliminary work presented in [22] only considers blind signature schemes.) We also notice that Helios 2.0 does not guarantee eligibility verifiability and is vulnerable to ballot stuffing by dishonest administrators.

Related work. Juels *et al.* [17, 18] present a definition of universal verifiability in the provable security model. Their definition assumes voting protocols produce non-interactive zero-knowledge proofs demonstrating the correctness of tallying. Here we consider definitions in a symbolic model. Universal verifiability was also studied by Chevallier-Mames *et al.* [11]. They show an incompatibility result: protocols cannot satisfy verifiability and vote privacy in an unconditional way (without relying on computational assumptions). But as witnessed by [17, 18], weaker versions of these properties can hold simultaneously. Our case studies demonstrate that our definition allows privacy and verifiability to coexist (see [14, 6] for a study of privacy properties in the applied pi calculus). Baskar *et al.* [7] and subsequently Talbi *et al.* [23] formalised individual and universal verifiability for the protocol by Fujioka *et al.* [15]. Their definitions are tightly coupled to that particular protocol and cannot easily be generalised. Moreover, their definitions characterise individual executions as verifiable or not; such properties should be considered with respect to every execution.

In our earlier work [22] a preliminary definition of election verifiability was presented with support for automated reasoning. However, that definition is too strong to hold on protocols such as [18, 4]. In particular, our earlier definition was only illustrated on a simplified version of [18] which did not satisfy coercion-resistance because we omitted the mixnets. Hence, this is the first general, symbolic definition which can be used to show verifiability for many important protocols, such as the ones studied in this paper.

2 Applied pi calculus

The calculus assumes an infinite set of names $a, b, c, k, m, n, s, t, \ldots$, an infinite set of variables $v, x, y, z, \ldots$ and a finite signature Σ, that is, a finite set of function symbols each with an associated arity. We also assume an infinite set of *record variables* $r, r_1, \ldots$. A function symbol of arity 0 is a constant. We use metavariables u, w to range over both names and variables. Terms L, M, N, T, U, V are built by applying function symbols to names, variables and other terms. Tuples $u_1, \ldots, u_l$ and $M_1, \ldots, M_l$ are occasionally abbreviated $\tilde{u}$ and $\tilde{M}$. We write $\{M_1/x_1, \ldots, M_l/x_l\}$ for substitutions that replace variables $x_1, \ldots, x_l$ with terms $M_1, \ldots, M_l$.

The applied pi calculus [1, **?**] relies on a simple sort system. Terms can be of sort Channel for channel names or Base for the payload sent out on these channels. Function symbols can only be applied to, and return, terms of sort Base. A term is ground when it does not contain variables.

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

$P, Q, R ::=$	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		
$\mathsf{rec}(r, M).P$	record message		
if $M = N$ then P else Q	conditional		

Fig. 1. Applied pi calculus grammar

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

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

The sets of free and bound names and variables in process A are denoted by $\mathrm{fn}(A)$, $\mathrm{bn}(A)$, $\mathrm{fv}(A)$, $\mathrm{bv}(A)$. Similarly, we write $\mathrm{fn}(M)$, $\mathrm{fv}(M)$ for the names and variables in term M and $\mathrm{rv}(A)$ and $\mathrm{rv}(M)$ for the set of record variables in a process and a term. An extended process A is *closed* if $\mathrm{fv}(A) = \emptyset$. A *context* $C[\_]$ is an extended process with a hole. 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. In this paper we tacitly assume that all signatures and equational theories contain the function symbols $\mathsf{pair}(\cdot,\cdot), \mathsf{fst}(\cdot), \mathsf{snd}(\cdot)$ and equations for pairing:

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

as well as some constant $\bot$. As a convenient shortcut we then write $(\!|T_1, \ldots T_n|\!)$ for $\mathsf{pair}(T_1, \mathsf{pair}(\ldots, \mathsf{pair}(T_n, \bot)))$ and $\pi_i(T)$ for $\mathsf{fst}(\mathsf{snd}^{i-1}(T))$.

Semantics. The operational semantics of the applied pi calculus are defined with respect to the three relations: structural equivalence ($\equiv$), internal reductions ($\rightarrow$) and labelled reduction ($\xrightarrow{\alpha}$). These semantics are standard and defined in [19]. We only illustrate them on an example (Figure 2). We write $\Longrightarrow$ for $(\rightarrow^* \xrightarrow{\alpha} \rightarrow^*)^*$, that is, the reflexive transitive closure of the labelled reduction.

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

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

Observe that labelled outputs are done by reference and extend the domain of the process's frame.

Fig. 2. A sequence of reductions in the applied pi semantics

3 Formalising voting protocols

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

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

For the purposes of individual verifiability the voter may rely on some data derived during the protocol execution. We keep track of all such values using the record construct (Definition 2).

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

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

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

In the above definition odd and even are used as a convenient way to split an infinite list into two infinite lists.

Given a sequence of record variables $\tilde{r}$, we denote by $\tilde{r}_i$ the sequence of variables obtained by indexing each variable in $\tilde{r}$ with i. A voting process can now be constructed such that the voter V records the values constructed and input during execution.

Definition 3. *Given a voting process specification $\langle V, A\rangle$, integer $n \in \mathbb{N}$, and names $s_1, \ldots, s_n$, we build the augmented voting process $\mathsf{VP}^+_n(s_1, \ldots, s_n) = A[V_1^+ \mid \cdots \mid V_n^+]$ where $V_i^+ = \mathsf{R}(V)\{^{s_i}/_v\}\{^{r_i}/_r \mid r \in rv(\mathsf{R}(V))\}$.*

The process $\mathsf{VP}^+_n(s_1, \ldots, s_n)$ models the voting protocol for n voters casting votes $s_1, \ldots, s_n$, who privately record the data that may be needed for verification using record variables $\tilde{r}_i$.

4 Election verifiability

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

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

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

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

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

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

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

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

4.1 Individual and universal verifiability

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

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

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

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

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

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

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

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

An individual voter should verify that the test Φ^{IV} holds when instantiated with her vote s_i, the information $\tilde{r}_i$ recorded during the execution of the protocol and some bulletin board entry. Indeed, Condition (1) ensures that the test will hold for at most one bulletin board entry. (Note that Φ^{IV}_i and Φ^{IV}_j are evaluated with the same ballot $y\sigma$ provided by $C[\_]$.) The fact that her ballot is counted will be ensured by Φ^{UV} which should also be tested by the voter. An observer will instantiate the test Φ^{UV} with the bulletin board entries $\tilde{y}$ and the declared outcome $\tilde{v}$. Condition (2) ensures the observer that Φ^{UV} only holds for a single outcome. Condition (3) ensures that if a bulletin board contains the ballots of voters who voted $s_1, \ldots, s_n$ then Φ^{UV} only holds if the declared outcome is (a permutation of) these votes. Finally, Condition (4) ensures that there exists an execution where the tests hold. In particular this allows us to verify whether the protocol can satisfy the tests when executed as expected. This also avoids tests which are always false and would make Conditions (1)-(3) vacuously hold.

4.2 Case study: FOO

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

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

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

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

Model in applied pi. The parts of the protocol that need to be trusted for achieving verifiability are surprisingly simple (Definition 5). The name rnd models the randomness that is supposed to be used to compute the commitment of the vote. All a voter needs to ensure is that the randomness used for the commitment is fresh. To ensure verifiability, all other operations such as computing the commitment, blinding and signing can be performed by the untrusted terminal.

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

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

Individual and universal verifiability. We define the tests

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

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

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

4.3 Case study: Helios 2.0

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

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

1. To cast a vote, the user runs a browser script that inputs her vote and creates a ballot that is encrypted with the public key of the election. The ballot includes a ZKP that the ballot represents an allowed vote (this is needed because the ballots are never decrypted individually).
2. The user can audit the ballot to check if it really represents a vote for her chosen candidate; if she elects to do this, the script provides her with the random data used in the ballot creation. She can then independently verify that the ballot was correctly constructed, but the ballot is now invalid and she has to create another one.
3. When the voter has decided to cast her ballot, the voter's browser submits it along with her pseudo-identity to the server. The server checks the ZKPs of the ballots, and publishes them on a bulletin board.
4. Individual voters can check that their ballots appear on the bulletin board. Any observer can check that the ballots that appear on the bulletin board represent allowed votes, by checking the ZKPs.
5. The server homomorphically combines the ballots, and publishes the encrypted tally. Anyone can check that this tally is done correctly.
6. The server submits the encrypted tally to each of the trustees, and obtains their share of the decryption key for that particular ciphertext, together with a proof that the key share is well-formed. The server publishes these key shares along with the proofs. Anyone can check the proofs.
7. The server decrypts the tally and publishes the result. Anyone can check this decryption.

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

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

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

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

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

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

Note that in the equation for $\mathsf{checkBallotPf}$ s is a name and not a variable. As the equational theory is closed under bijective renaming of names this equation holds for any name, but fails if one replaces the name by a term, *e.g.*, $s + s$. We suppose that all names are possible votes but give the possibility to check that a voter does not include a term $s + s$ which would add a vote to the outcome.

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

- The script that constructs the ballot. Although the voter cannot verify it, the trust in this script is motivated by the fact that she is able to audit it.
- The trustees. Although the trustees' behaviour cannot be verified, voters and observers may want to trust them because trust is distributed among them.

Hence, we include these two components in the context A_{helios} of our voting process specification.

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

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

We suppose that the inputs of x_{pid}, x_{ballot} and x_{ballotpf} are stored in record variables r_{pid}, r_{ballot} and $r_{ballotpf}$ respectively. The voter V_{helios} receives her voter id pid on a private channel. She sends her vote on the channel to A_{helios}, which creates the ballot for her. She receives the ballot and sends it (paired with pid) to the server. A_{helios} represents the parts of the system that are required to be trusted. It publishes the election key and issues voter ids. It includes the ballot creation script B, which receives a voter's vote, creates a random m and forms the ballot, along with its proof, and returns it to the voter. A_{helios} also contains the trustee T, which accepts a tally ciphertext and returns a decryption key for it, along with the proof that the decryption key is correct. We assume the trustee will decrypt any ciphertext (but only one).

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

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

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

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

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

5 Eligibility verifiability

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

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

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

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

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

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

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

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

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

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

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

5.1 Case study: JCJ-Civitas

The protocol due to Juels *et al.* [18] is based on mixnets and was implemented by Clarkson *et al.* [13, 12] as an open-source voting system called Civitas.

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

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

The registration phase then proceeds as follows.

2. The registrars generate and distribute voter credentials: a private part d and a public part $\mathsf{penc}(\mathsf{pk}(sk_R), m'', d)$ (the probabilistic encryption of d under the registrars' public key $\mathsf{pk}(sk_R)$). This is done in a distributed manner, so that no individual registrar learns the value of any private credential d.
3. The registrars publish the signed public voter credentials.
4. The registrars announce the candidate list $\tilde{t} = (t_1, \ldots, t_l)$.

The protocol then enters the voting phase.

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

After some predefined deadline the tallying phase commences.

6. The talliers read the n' ballots posted to the bulletin board by voters (that is, the triples consisting of the two ciphertexts and the zero-knowledge proof) and discards any entries for which the zero-knowledge proof does not hold.
7. The elimination of re-votes is performed on the ballots using pairwise *plaintext equality tests* (PET) on the ciphertexts containing private voter credentials. (A PET [16] is a cryptographic predicate which allows a keyholder to provide a proof that two ciphertexts contain the same plaintext.) Re-vote elimination is performed in a verifiable manner with respect to some publicly defined policy, *e.g.*, by the order of ballots on the bulletin board.
8. The talliers perform a verifiable re-encryption mix on the ballots (ballots consist of a vote ciphertext and a public credential ciphertext; the link between both is preserved by the mix.) The mix ensures that a voter cannot trace her vote, allowing the protocol to achieve coercion-resistance.
9. The talliers perform a verifiable re-encryption mix on the list of public credentials published by the registrars. This mix anonymises public voter credentials, breaking any link with the voter for privacy purposes.
10. Ballots based on invalid credentials are weeded using PETs between the mixed ballots and the mixed public credentials. Both have been posted to the bulletin board. (Using PETs the correctness of weeding is verifiable.)
11. Finally, the talliers perform a verifiable decryption and publish the result.

Equational theory. The protocol uses a variant of the ElGamal encryption scheme [18]. Accordingly we adopt the signature and associated equational theory from the Helios case study. We model the ZK proof demonstrating correct construction of the voter's ciphertexts, re-encryption and PETs by the equations

$$\begin{aligned}
&\mathsf{checkBallot}(\mathsf{ballotPf}(x_{\mathsf{pk}}, x_{\mathsf{rand}}, x_{\mathsf{text}}, x'_{\mathsf{pk}}, x'_{\mathsf{rand}}, x'_{\mathsf{text}}),\\
&\qquad\qquad \mathsf{penc}(x_{\mathsf{pk}}, x_{\mathsf{rand}}, x_{\mathsf{text}}), \mathsf{penc}(x'_{\mathsf{pk}}, x'_{\mathsf{rand}}, x'_{\mathsf{text}})) = \mathsf{true}\\
&\mathsf{renc}(y_{\mathsf{rand}}, \mathsf{penc}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{rand}}, x_{\mathsf{text}})) = \mathsf{penc}(\mathsf{pk}(x_{\mathsf{sk}}), f(x_{\mathsf{rand}}, y_{\mathsf{rand}}), x_{\mathsf{text}})\\
&\mathsf{pet}(\mathsf{petPf}(x_{\mathsf{sk}}, ciph, ciph'), ciph, ciph') = \mathsf{true}
\end{aligned}$$

where $ciph \mathrel{\hat{=}} \mathsf{penc}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{rand}}, x_{\mathsf{text}})$ and $ciph' \mathrel{\hat{=}} \mathsf{penc}(\mathsf{pk}(x_{\mathsf{sk}}), x'_{\mathsf{rand}}, x_{\mathsf{text}})$. In addition we consider verifiable re-encryption mixnets and introduce for each permutation χ on $\{1, \ldots, n\}$ the equation:

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

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

$$\begin{aligned}
&\mathsf{checkMixPair}(\mathsf{mixPairPf}(\langle\!| x_1, x'_1 |\!\rangle, \ldots, \langle\!| x_n, x'_n |\!\rangle, \langle\!| c_1, c'_1 |\!\rangle, \ldots, \langle\!| c_n, c'_n |\!\rangle,\\
&\quad \langle\!| z_1, z'_1 |\!\rangle, \ldots, \langle\!| z_n, z'_n |\!\rangle), \langle\!| x_1, x'_1 |\!\rangle, \ldots, \langle\!| x_n, x'_n |\!\rangle, \langle\!| c_1, c'_1 |\!\rangle, \ldots, \langle\!| c_n, c'_n |\!\rangle) = \mathsf{true}
\end{aligned}$$

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

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

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

Although neither voters nor observers can verify that the registrars adhere to such expectations, they trust them because trust is distributed. The trusted components are modelled by the voting process specification $\langle A_{\mathsf{jcj}}, V_{\mathsf{jcj}}\rangle$ (Definition 8). The context A_{jcj} distributes private keys on a private channel, launches an unbounded number of registrar processes and publishes the public keys of both the registrars and talliers. The registrar R constructs a fresh private credential d and sends the private credential along with the signed public part (that is, $\mathsf{sign}(ssk_R, \mathsf{penc}(x_{pk_R}, m'', d))$) to the voter; the registrar also publishes the signed public credential on the bulletin board. The voter V_{jcj} receives the private and public credentials from the registrar and constructs her ballot; that is, the pair of ciphertexts and a zero-knowledge proof demonstrating their correct construction.

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

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

Election verifiability. We suppose the recording function uses record variables $\tilde{r} = (r_{cred}, r_m, r_{m'}) = \mathrm{rv}(\mathsf{R}(V))$ (corresponding to the variable x_{cred} and names m, m' in the process V). Accordingly, given $n \in \mathbb{N}$ we define:

$$
\begin{aligned}
\Phi^{IV} &\mathrel{\hat{=}} y =_E (\!|\mathsf{penc}(x_{\mathsf{pk_T}}, r_m, v), \mathsf{penc}(x_{\mathsf{pk_R}}, r_{m'}, \pi_1(r_{cred})),\\
&\qquad\qquad \mathsf{ballotPf}(x_{\mathsf{pk_T}}, r_m, v, x_{\mathsf{pk_R}}, r_{m'}, \pi_1(r_{cred}))|\!) \wedge w = \pi_2(r_{cred})\\
\Phi^{UV} &\mathrel{\hat{=}} \mathsf{checkMixPair}(z_{\mathsf{mixPairPf}}, (\!|\pi_1(y_1), \pi_2(y_1)|\!), \ldots, (\!|\pi_1(y_n), \pi_2(y_n)|\!),\\
&\qquad\qquad z_{\mathsf{bal},1}, \ldots, z_{\mathsf{bal},n}) =_E \mathsf{true}\\
&\quad \wedge \textstyle\bigwedge_{i=1}^{n} \mathsf{dec}(z_{\mathsf{decKey},i}, \pi_1(z_{\mathsf{bal},i})) =_E v_i\\
&\quad \wedge \textstyle\bigwedge_{i=1}^{n} \mathsf{checkDecKeyPf}(x_{\mathsf{pk_T}}, \pi_1(z_{\mathsf{bal},i}), z_{\mathsf{decKey},i}, z_{\mathsf{decPf},i}) =_E \mathsf{true}\\
\Phi^{EV} &\mathrel{\hat{=}} \textstyle\bigwedge_{i=1}^{n} \mathsf{checkBallot}(\pi_3(y_i), \pi_1(y_i), \pi_2(y_i))\\
&\quad \wedge \mathsf{checkMixPair}(z_{\mathsf{mixPairPf}}, (\!|\pi_1(y_1), \pi_2(y_1)|\!), \ldots, (\!|\pi_1(y_n), \pi_2(y_n)|\!),\\
&\qquad\qquad z_{\mathsf{bal},1}, \ldots, z_{\mathsf{bal},n}) =_E \mathsf{true}\\
&\quad \wedge \textstyle\bigwedge_{i=1}^{n} \mathsf{pet}(z_{\mathsf{petPf},i}, \pi_2(z_{\mathsf{bal},i}), \hat{z}_{\mathsf{cred},i}) =_E \mathsf{true}\\
&\quad \wedge (\!|z_{\mathsf{cred},1}, \ldots, z_{\mathsf{cred},n}|\!) \simeq (\!|\hat{z}_{\mathsf{cred},1}, \ldots, \hat{z}_{\mathsf{cred},n}|\!)\\
&\quad \wedge \mathsf{checkMix}(z_{\mathsf{mixPf}}, \mathsf{getmsg}(w_1), \ldots, \mathsf{getmsg}(w_n), z_{\mathsf{cred},1}, \ldots, z_{\mathsf{cred},n}) =_E \mathsf{true}\\
&\quad \wedge \textstyle\bigwedge_{i=1}^{n} \mathsf{checksign}(x_{spk_R}, w_i)
\end{aligned}
$$

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

6 Conclusion

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

Acknowledgements

We are particularly grateful to Michael Clarkson for careful reading of our preliminary formal definition of election verifiability. His comments provided useful guidance for the definition we present here.

References

1. M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *POPL'01: Proc. 28th ACM Symposium on Principles of Programming Languages*, pages 104–115, New York, USA, 2001. ACM.
2. B. Adida. *Advances in Cryptographic Voting Systems*. PhD thesis, MIT, 2006.
3. B. Adida. Helios: Web-based open-audit voting. In *Proc. 17th 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: Proc. 21st IEEE Computer Security Foundations Symposium*, pages 195–209, Washington, USA, 2008. IEEE.
7. A. Baskar, R. Ramanujam, and S. P. Suresh. Knowledge-based modelling of voting protocols. In *TARK'07: Proc. 11th International Conference on Theoretical Aspects of Rationality and Knowledge*, pages 62–71. ACM, 2007.
8. D. Bowen. Secretary of State Debra Bowen Moves to Strengthen Voter Confidence in Election Security Following Top-to-Bottom Review of Voting Systems. California Secretary of State, press release DB07:042 `http://www.sos.ca.gov/elections/voting_systems/ttbr/db07_042_ttbr_system_decisions_release.pdf`, August 2007.

9. Bundesverfassungsgericht (Germany's Federal Constitutional Court). Use of voting computers in 2005 Bundestag election unconstitutional. Press release 19/2009 http://www.bundesverfassungsgericht.de/en/press/bvg09-019en.html, March 2009.
10. D. Chaum, P. Y. A. Ryan, and S. Schneider. A practical, voter-verifiable election scheme. In *ESORICS'05: Proc. 10th European Symposium On Research In Computer Security*, volume 3679 of *LNCS*, pages 118–139. Springer, 2005.
11. B. Chevallier-Mames, P.-A. Fouque, D. Pointcheval, J. Stern, and J. Traore. On Some Incompatible Properties of Voting Schemes. In *WOTE'06: Proc. Workshop on Trustworthy Elections*, 2006.
12. M. R. Clarkson, S. Chong, and A. C. Myers. Civitas: Toward a secure voting system. Technical Report 2007-2081, Cornell University, May 2007. Revised March 2008. http://hdl.handle.net/1813/7875.
13. M. R. Clarkson, S. Chong, and A. C. Myers. Civitas: Toward a secure voting system. In *S&P'08: Proc. Symposium on Security and Privacy*, pages 354–368. IEEE Computer Society, 2008.
14. S. Delaune, S. Kremer, and M. D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, July 2009.
15. A. Fujioka, T. Okamoto, and K. Ohta. A Practical Secret Voting Scheme for Large Scale Elections. In *ASIACRYPT'92: Proc. Workshop on the Theory and Application of Cryptographic Techniques*, pages 244–251. Springer, 1992.
16. M. Jakobsson and A. Juels. Mix and match: Secure function evaluation via ciphertexts. In *ASIACRYPT'00: Proc. 6th International Conference on the Theory and Application of Cryptology and Information Security*, pages 162–177. Springer, 2000.
17. A. Juels, D. Catalano, and M. Jakobsson. Coercion-Resistant Electronic Elections. Cryptology ePrint Archive, Report 2002/165, 2002.
18. A. Juels, D. Catalano, and M. Jakobsson. Coercion-resistant electronic elections. In *WPES '05: Proc. workshop on Privacy in the electronic society*, pages 61–70. ACM, 2005. See also http://www.rsa.com/rsalabs/node.asp?id=2860.
19. S. Kremer, B. Smyth, and M. D. Ryan. Election verifiability in electronic voting protocols. Technical Report CSR-10-06, University of Birmingham, School of Computer Science, 2010. Available at http://www.bensmyth.com/publications/10tech/CSR-10-06.pdf.
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. Participants of the Dagstuhl Conference on Frontiers of E-Voting. Dagstuhl accord. http://www.dagstuhlaccord.org/, 2007.
22. B. Smyth, M. D. Ryan, S. Kremer, and M. Kourjieh. Towards automatic analysis of election verifiability properties. In *Joint Workshop on Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security (ARSPA-WITS'10)*, LNCS. Springer, 2010. To appear.
23. 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: Proc. 10th International Conference on Information and Communications Security Conference*, pages 403–418, London, 2008. Springer.
24. UK Electoral Commission. Key issues and conclusions: May 2007 electoral pilot schemes. http://www.electoralcommission.org.uk/elections/pilots/May2007.