D4-1. Formal description of our case study: Helios 2.0

Ben Smyth and Véronique Cortier

Loria, CNRS, France

Abstract

Helios 2.0 is an open-source web-based end-to-end verifiable electronic voting system, suitable for use in low-coercion environments. In this report, we present a cryptographic description of the Helios protocol and a model in the applied pi calculus, suited to the analysis of privacy.

1 Introduction

Paper-based elections derive privacy properties from physical characteristics of the real-world, for example, the indistinguishability of an individual's ballot from an arbitrary ballot, and the inability of a coercer to collaborate with a voter inside a polling booth. Similarly, the physical world provides transparency by allowing observation of the whole process (that is, from ballot casting to tallying) and robustness characteristics, for example, by ensuring that the markings on paper ballot sealed inside a locked ballot box cannot be altered. Replicating these attributes in a digital setting has proven to be difficult and, hence, is an active research topic [1, 2, 3].

Informally, privacy [4, 5, 6] and verifiability [3, 7, 8] properties for electronic voting systems are characterised as follows.

Ballot secrecy. A voter's vote is not revealed to anyone.

Receipt freeness. A voter cannot gain information which can be used to prove, to a coercer, how she voted.

Coercion resistance. A voter cannot collaborate, with a coercer, to gain information which can be used to prove how she voted.

Individual verifiability. A voter can check that her own ballot is published on the election's bulletin board.

Universal verifiability. Anyone can check that all the votes in the election outcome correspond to ballots published on the election's bulletin board.

The verifiability properties (also called *end-to-end verifiability* [3, 9, 10, 7, 11]) 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, thereby simulating the transparency and robustness characteristics found in paper-based elections.

Helios is an open-source web-based electronic voting system. The scheme is claimed to satisfy ballot secrecy [12], but the nature of remote voting makes the possibility of satisfying stronger privacy properties difficult, and Helios does not satisfy receipt freeness nor coercion resistance. In addition to ballot secrecy, the system provides end-to-end verifiability (cf. [8, 13] and [14, Chapter 3] for an analysis of end-to-end verifiability in Helios). Helios is particularly significant due to its real-world deployment: the International Association of Cryptologic Research used Helios to elect its board members [15], following a successful trial in a non-binding poll [16]; the Catholic University of Louvain adopted the system to elect the university president [12]; and Princeton University used Helios to elect the student vice president [17].

Contribution of this report. In this report we present a cryptographic description of the Helios protocol and a model in the applied pi calculus which is suitable for the analysis of ballot secrecy.

2 Helios 2.0

Helios exploits the additive homomorphic [18, 19, 20] and distributed decryption [21, 22] properties of ElGamal [23]. We will recall these cryptographic details before presenting the Helios protocol.

2.1 Background: Additive homomorphic ElGamal

Given cryptographic parameters (p, q, g) and a number $n \in \mathbb{N}$ of trustees, where p and q are large primes such that $q \mid p-1$ and g is a generator of the multiplicative group $\mathbb{Z}_p^*$ of order q, the following operations are defined by ElGamal.

Distributed key generation. Each trustee $i \in n$ selects a private key share $x_i \in_R \mathbb{Z}_q^*$ and computes a public key share $h_i = g^{x_i} \bmod p$. The public key is $h = h_1 \cdot \ldots \cdot h_n \bmod p$.

Encryption. Given a message m and a public key h, select a random nonce $r \in_R \mathbb{Z}_q^*$ and derive the ciphertext $(a, b) = (g^r \bmod p,\ g^m \cdot h^r \bmod p)$.

Re-encryption. Given a ciphertext (a, b) and public key h, select a random nonce $r' \in_R \mathbb{Z}_q^*$ and derive the re-encrypted ciphertext $(a', b') = (a \cdot g^{r'} \bmod p,\ b \cdot h^{r'} \bmod p)$.

Homomorphic addition. Given two ciphertexts (a, b) and (a', b'), the homomorphic addition of plaintexts is computed by multiplication $(a \cdot a' \bmod p,\ b \cdot b' \bmod p)$.

Distributed decryption. Given a ciphertext (a, b), each trustee $i \in n$ computes the partial decryption $k_i = a^{x_i}$. The plaintext $m = \log_g M$ is recovered from $M = b/(k_1 \cdot \ldots \cdot k_n) \bmod p$.

The computation of a discrete logarithm $\log_g M$ is hard in general. However, if M is chosen from a restricted domain, then the complexity is reduced; for example, if M is an integer such that $0 \leq M \leq n$, then the complexity is $O(n)$ by linear search or $O(\sqrt{n})$ using the baby-step giant-step algorithm [24] (see also [25, §3.1]).

For secrecy, each trustee $i \in n$ must demonstrate knowledge of a discrete logarithm $\log_g h_i$, that is, they proof that h_i has been correctly constructed; this prevents, for example, a trustee constructing their public key share $h_i = h$. For integrity of decryption, each trustee $i \in n$ must demonstrate equality between discrete logarithms $\log_g h_i$ and $\log_a k_i$; this prevents, for example, a trustee constructing the public key share $h_i = g^{m+x_i}$ and providing the partial decryption $k_i = a^{x_i}$. In addition, the voter must demonstrate that a valid vote has been encrypted. These proofs can be achieved using signatures of knowledge (see Appendix A for details).

2.2 Protocol description

An election is created by naming an election officer, selecting a set of trustees, and generating a distributed public key pair. The election officer publishes, on the bulletin board, the public part of the trustees' key (and proof of correct construction), the candidate list $\tilde{t} = (t_1, \ldots, t_l) \cup \{\epsilon\}$ (where ϵ represents a vote of abstention), and the list of eligible voters $\widetilde{id} = (id_1, \ldots, id_n)$; the officer also publishes the *election fingerprint*, that is, the hash of these parameters. Informally, the steps that participants take during a run of Helios are as follows.

1. The voter launches a browser script that downloads the election parameters and recomputes the election fingerprint. The voter should verify that the fingerprint corresponds to the value published on the bulletin board. (This ensures that the script is using the trustees' public key; in particular, it helps prevent encrypting a vote with an adversary's public key. Such attacks have been discussed in the context of Direct Anonymous Attestation by Rudolph [26]; although, the vulnerability was discounted, in the trusted computing setting, by Leung, Chen & Mitchell [27].)

2. The voter inputs her vote $v \in \tilde{t}$ to the browser script, which creates a ballot consisting of her vote encrypted by the trustees' public key, and a proof that the ballot represents a permitted vote (this is needed because the ballots are never decrypted individually, in particular, it prevents multiple votes being encoded as a single ballot). The ballot is displayed to the voter.

3. The voter can audit the ballot to check if it really represents a vote for her chosen candidate; if she decides to do this, then the script provides her with the random data used in the ballot creation. She can then independently

reconstruct her ballot and verify that it is indeed well-formed, but the ballot is now invalid. (Invalidating audited ballots provides some practical resistance against vote selling.) See Benaloh [28, 29] for further details on ballot auditing.

4. When the voter has decided to cast her ballot, she submits it to the election officer. The election officer authenticates the voter and checks that she is eligible to vote. The election officer also verifies the proof and publishes the ballot, appended with the voter's identity id, on the bulletin board. (In practice, the election officer also publishes the hash of the ballot, we omit this detail for brevity.)

5. Individual voters can check that their ballots appear on the bulletin board and, by verifying the proof, observers are assured that ballots represent permitted votes.

6. After some predefined deadline, the election officer homomorphically combines the ballots and publishes the encrypted tally on the bulletin board. Anyone can check that tallying is performed correctly.

7. Each of the trustees publishes a partial decryption of the encrypted tally, together with a signature of knowledge proving the partial decryption's correct construction. Anyone can verify these proofs.

8. The election officer decrypts the tally and publishes the result. Anyone can check this decryption.

Formally, Step 1 is defined in Figure 1. (For simplicity the ballot construction algorithm in Figure 1 considers a vote $v \in \tilde{t}$, this can be generalised [12] to consider a vote $\tilde{v} \subseteq \tilde{t}$.) Checking voter eligibility (Step 4) is beyond the scope of Helios and [12] proposes the use of existing infrastructure. The remaining steps follow immediately from the application of cryptographic primitives (see Section 2.1 for details).

2.3 Software implementation

Helios 3.0 is an extension of Helios 2.0 which adds numerous practical features, including: integration of authentication with various web-services (for example, Facebook, GMail and Twitter), bulk voter registration using pre-existing electoral rolls, and simplification of administration with multiple trustees. Helios 3.0 has been implemented and is publicly available: `http://heliosvoting.org/`.

3 Formal model

In this section, we formally model the Helios 2.0 protocol using the applied pi calculus [30, 31]. (This suitability of this calculus for evaluating security properties of electronic voting protocols has previously been shown; see, for example, [32, 6, 8].)

Figure 1 Ballot construction by the browser script

Input: Cryptographic parameters (p, q, g), public key h, candidate list $\tilde{t} = (t_1, \ldots, t_l) \cup \{\epsilon\}$ and vote v.

Output: Encrypted vote $(a_1, b_1), \ldots, (a_l, b_l)$, signatures of knowledge $(\bar{a}_1, \bar{b}_1, \bar{c}_1, \bar{s}_1, \bar{a}'_1, \bar{b}'_1, \bar{c}'_1, \bar{s}'_1), \ldots, (\bar{a}_l, \bar{b}_l, \bar{c}_l, \bar{s}_l, \bar{a}'_l, \bar{b}'_l, \bar{c}'_l, \bar{s}'_l)$ and signature of knowledge $(\bar{a}, \bar{b}, \bar{c}, \bar{s}, \bar{a}', \bar{b}', \bar{c}', \bar{s}')$.

1. If $v \notin \tilde{t}$ then the script terminates.
2. Encode the vote v as a bitstring. For all $1 \leq i \leq l$, let
$$m_i = \begin{cases} 1 & \text{if } v = t_i \\ 0 & \text{otherwise} \end{cases}$$
3. The bitstring representing the vote is encrypted. For all $1 \leq i \leq l$, let
$$(a_i, b_i) = (g^{r_i} \bmod p,\ g^{m_i} \cdot h^{r_i} \bmod p)$$
where $r_i \in_R \mathbb{Z}_q^*$.
4. For all $1 \leq i \leq l$, let $(\bar{a}_i, \bar{b}_i, \bar{c}_i, \bar{s}_i, \bar{a}'_i, \bar{b}'_i, \bar{c}'_i, \bar{s}'_i)$ be a signature of knowledge demonstrating that the ciphertext (a_i, b_i) contains either 0 or 1, that is, each candidate can receive at most one vote.
5. Let $(\bar{a}, \bar{b}, \bar{c}, \bar{s}, \bar{a}', \bar{b}', \bar{c}', \bar{s}')$ be a signature of knowledge demonstrating that the ciphertext $(a_1 \cdot \ldots \cdot a_l,\ b_1 \cdot \ldots \cdot b_l)$ contains either 0 or 1, that is, at most one candidate receives one vote.

3.1 Applied pi calculus

We first recall the applied pi calculus setting [30]. We assume an infinite set of *names* $a, b, c, \ldots, k, \ldots, m, n, \ldots, s, \ldots$, an infinite set of *variables* $x, y, z, \ldots$, and a *signature* Σ consisting of a finite set of *function symbols*, each with an associated arity. 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. We write $\{^M/_x\}$ for the *substitution* that replaces the variable x with the term M. Arbitrarily large substitutions can be written as $\{^{M_1}/_{x_1}, \ldots, ^{M_l}/_{x_l}\}$ and the letters σ and τ range over substitutions. We write $N\sigma$ for the result of applying σ to the free variables of term N. A term is *ground* when it does not contain variables.

The signature Σ is equipped with an *equational theory* E, that is, a set of equations of the form $M = N$, where the terms M, N are defined over the signature Σ. We define equality modulo the equational theory, written $=_E$, as the smallest equivalence relation on terms that contains E and is closed under

Figure 2 Syntax for processes

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

application of function symbols, substitution of terms for variables and bijective renaming of names. We write $M =_E N$ when the equation $M = N$ is in the theory E, and keep the signature implicit. When E is clear from its usage, we may abbreviate $M =_E N$ as $M = N$. The negation of $M =_E N$ is denoted $M \neq_E N$ (and similarly abbreviated $M \neq N$).

Processes and *extended processes* are defined in the usual way (Figure 2). We write $\nu\, \tilde{u}$ for the (possibly empty) series of pairwise-distinct binders $\nu\, u_1. \cdots .\nu\, u_l$. The active substitution $\{^M/_x\}$ can replace the variable x for the term M in every process it comes into contact with and this behaviour can be controlled by restriction, in particular, the process $\nu\, x.(\{^M/_x\} \mid P)$ corresponds exactly to let $x = M$ in P. Arbitrarily large active substitutions can be obtained by parallel composition and we occasionally abbreviate $\{^{M_1}/_{x_1}\} \mid \ldots \mid \{^{M_l}/_{x_l}\}$ as $\{^{M_1}/_{x_1}, \ldots, ^{M_l}/_{x_l}\}$ or $\{^{\tilde{M}}/_{\tilde{x}}\}$. We also use σ and τ to range over active substitutions, and write $N\sigma$ for the result of applying σ to the free variables of N. Extended processes must have at most one active substitution for each variable and there is exactly one when the variable is under restriction. The only minor change compared to [30] is that conditional branches now depend on formulae $\phi, \psi ::= M = N \mid M \neq N \mid \phi \wedge \psi$. If M and N are ground, we define $[\![M = N]\!]$ to be true if $M =_E N$ and false otherwise. The semantics of $[\![\ ]\!]$ is then extended to formulae in the standard way.

The *scope* of names and variables are delimited by binders $u(x)$ and $\nu\, u$. The set of bound names is written $\mathrm{bn}(A)$ and the set of bound variables is written $\mathrm{bv}(A)$; similarly we define the set of free names $\mathrm{fn}(A)$ and free variables $\mathrm{fv}(A)$. Occasionally, we write $\mathrm{fn}(M)$ (and $\mathrm{fv}(M)$ respectively) for the set of names (and respectively variables) which appear in term M. An extended process is *closed* when every variable x is either bound or defined by an active substitution.

We define a *context* $C[\_]$ to be an extended process with a hole. We obtain $C[A]$ as the result of filling $C[\_]$'s hole with the extended process A. An *evaluation context* is a context whose hole is not in the scope of a replication, a conditional, an input, or an output. A context $C[\_]$ closes A when $C[A]$ is closed.

A *frame*, denoted φ or ψ, is an extended process built from the null process 0 and active substitutions $\{^M/_x\}$, which are composed by parallel composition and restriction. The *domain* $\mathrm{dom}(\varphi)$ of a frame φ is the set of variables that φ exports, that is, the set of variables x for which φ contains an active substitution $\{^M/_x\}$ such that x is not under restriction. Every extended process A can be mapped to a frame $\varphi(A)$ by replacing every plain process in A with 0.

3.1.1 Operational semantics

The operational semantics are defined by three relations: *structural equivalence* ($\equiv$), *internal reduction* ($\rightarrow$), and *labelled reduction* ($\xrightarrow{\alpha}$). These relations satisfy the rules in Figure 3 and are defined such that: structural equivalence is the smallest equivalence relation on extended processes that is closed by α-conversion of both bound names and bound variables, and closed under application of evaluation contexts; internal reduction is the smallest relation on extended processes closed under structural equivalence and application of evaluation contexts; and for labelled reductions α is a *label* of the form $c(M)$, $\overline{c}\langle u\rangle$, or $\nu\, u.\overline{c}\langle u\rangle$ such that u is either a channel name or a variable of base type.

3.1.2 Equivalence

The definition of observational equivalence [30] quantifies over all contexts which makes proofs difficult, therefore we adopt labelled bisimilarity in this paper. Labelled bisimilarity relies on an equivalence relation between frames, called static equivalence.

Definition 1 (Static equivalence) *Two closed frames φ and ψ are statically equivalent, denoted $\varphi \approx_s \psi$, if $\mathrm{dom}(\varphi) = \mathrm{dom}(\psi)$ and there exists a set of names $\tilde{n}$ and substitutions σ, τ such that $\varphi \equiv \nu\, \tilde{n}.\sigma$ and $\psi \equiv \nu\, \tilde{n}.\tau$ and for all terms M, N such that $\tilde{n} \cap (\mathrm{fn}(M) \cup \mathrm{fn}(N)) = \emptyset$, we have $M\sigma =_E N\sigma$ holds if and only if $M\tau =_E N\tau$ holds. Two closed extended processes A, B are statically equivalent, written $A \approx_s B$, if their frames are statically equivalent; that is, $\varphi(A) \approx_s \varphi(B)$.*

The relation $\approx_s$ is called *static* equivalence because it only examines the current state of the processes, and not the processes' dynamic behaviour. The following definition of labelled bisimilarity captures the dynamic part.

Definition 2 (Labelled bisimilarity) *Labelled bisimilarity ($\approx_l$) is the largest symmetric relation $\mathcal{R}$ on closed extended processes such that $A\, \mathcal{R}\, B$ implies:*

1. *$A \approx_s B$;*
2. *if $A \rightarrow A'$, then $B \rightarrow^* B'$ and $A'\, \mathcal{R}\, B'$ for some B';*

Figure 3 Semantics for processes

$$
\begin{array}{lrcl}
\textsc{Par-0} & A & \equiv & A \mid 0 \\
\textsc{Par-A} & A \mid (B \mid C) & \equiv & (A \mid B) \mid C \\
\textsc{Par-C} & A \mid B & \equiv & B \mid A \\
\textsc{Repl} & !P & \equiv & P \mid !P
\end{array}
$$

$$
\begin{array}{lrcl}
\textsc{New-0} & \nu\, n.0 & \equiv & 0 \\
\textsc{New-C} & \nu\, u.\nu\, w.A & \equiv & \nu\, w.\nu\, u.A \\
\textsc{New-Par} & A \mid \nu\, u.B & \equiv & \nu\, u.(A \mid B) \\
 & & & \text{where } u \notin \mathrm{fv}(A) \cup \mathrm{fn}(A)
\end{array}
$$

$$
\begin{array}{lrcl}
\textsc{Alias} & \nu\, x.\{^M/_x\} & \equiv & 0 \\
\textsc{Subst} & \{^M/_x\} \mid A & \equiv & \{^M/_x\} \mid A\{^M/_x\} \\
\textsc{Rewrite} & \{^M/_x\} & \equiv & \{^N/_x\} \\
 & & & \text{where } M =_E N
\end{array}
$$

$$
\begin{array}{ll}
\textsc{Comm} & \overline{c}\langle x\rangle.P \mid c(x).Q \to P \mid Q \\
\textsc{Then} & \text{if } \phi \text{ then } P \text{ else } Q \to P \quad \text{if } [\![\phi]\!] = \mathsf{true} \\
\textsc{Else} & \text{if } \phi \text{ then } P \text{ else } Q \to Q \quad \text{otherwise}
\end{array}
$$

$$\textsc{In} \quad c(x).P \xrightarrow{c(M)} P\{^M/_x\}$$

$$\textsc{Out-Atom} \quad \overline{c}\langle u\rangle.P \xrightarrow{\overline{c}\langle u\rangle} P$$

$$\textsc{Open-Atom} \quad \frac{A \xrightarrow{\overline{c}\langle u\rangle} A' \qquad u \neq c}{\nu\, u.A \xrightarrow{\nu\, u.\overline{c}\langle u\rangle} A'}$$

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

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

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

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

3.2 Modelling Helios in applied pi

In the applied pi calculus, it is sufficient to model the parts of the voting system which need to be trusted for a particular security property. Ultimately, our model will be used to consider ballot secrecy, so we shall consider the trusted components required for this property; all the remaining parts of the system are controlled by the adversarial environment. Accordingly, we assume the existence of at least two honest voters $\mathcal{A}$, $\mathcal{B}$; since this avoids the scenario where ballot secrecy of an individual voter is compromised by collusion amongst all the remaining voters. In addition, the following trust assumptions are required.

- At least one trustee is honest
- The election officer runs the bulletin board honestly:
  - Voters $\mathcal{A}$, $\mathcal{B}$ have authentic channels with the bulletin board
  - Signatures of knowledge are checked for dishonest voters*
  - Replays of honest ballots (that is, those cast by $\mathcal{A}$ or $\mathcal{B}$) are rejected*
  - The tally is correctly computed*
  - The trustees have an authentic channel with the bulletin board
- The browser script is trusted and has the correct public key of the election

(Assumptions marked with * could be performed by an honest trustee, rather than the bulletin board.) Although neither voters nor observers can verify that there exists an honest trustee, an assurance of trust is provided by distribution. The necessity to trust the election officer to run the bulletin board is less desirable and work-in-progress [33] aims to weaken this assumption; moreover, to further distribute trust assumptions, the trustees could also check signatures and tallying. Finally, trust in the browser script can be obtained by using software written by a reputable source or writing your own code.

The Helios voting system is modelled in Section 3.2.2 and in Section 3.2.1 we construct a suitable signature Σ to capture the cryptographic primitives used by the scheme and define an equational theory E to capture the relationship between these primitives.

3.2.1 Signature

We adopt the following signature.

$$\Sigma = \{\mathsf{ok}, \mathsf{zero}, \mathsf{one}, \bot, \mathsf{fst}, \mathsf{snd}, \mathsf{pair}, *, +, \circ, \mathsf{partial}, \mathsf{checkspk}, \mathsf{penc}, \mathsf{spk}, \}$$

Functions $\mathsf{ok}, \mathsf{zero}, \mathsf{one}, \bot$ are constants; $\mathsf{fst}, \mathsf{snd}$ are unary functions; $\mathsf{dec}, \mathsf{pair}, \mathsf{partial}, *, +, \circ$ are binary functions; $\mathsf{checkspk}, \mathsf{penc}$ are ternary functions; and spk is a function of arity four. We adopt infix notation for $*, +$, and $\circ$.

The term $\mathsf{penc}(T, N, M)$ denotes the encryption of plaintext M, using random nonce N and key T. The term $U{*}U'$ denotes the homomorphic combination of ciphertexts U and U', the corresponding operation on plaintexts is written $M + M'$ and $N \circ N'$ on nonces. The partial decryption of ciphertext U using key L is denoted $\mathsf{partial}(L, U)$. The term $\mathsf{spk}(T, N, M, U)$ represents a signature of knowledge that proves U is a ciphertext on the plaintext M using nonce N and such that M is either the constant zero or one. We introduce tuples using pairings and, for convenience, $\mathsf{pair}(M_1, \mathsf{pair}(\ldots, \mathsf{pair}(M_n, \bot)))$ is occasionally abbreviated as $(\!|M_1, \ldots, M_n|\!)$, and $\mathsf{fst}(\mathsf{snd}^{i-1}(M))$ is denoted $\pi_i(M)$, where $i \in \mathbb{N}$. We use the equational theory E that asserts functions $+, *, \circ$ are commutative and associative, and includes the equations:

$$\mathsf{fst}(\mathsf{pair}(x, y)) = x \tag{1}$$

$$\mathsf{snd}(\mathsf{pair}(x, y)) = y \tag{2}$$

$$\mathsf{zero} + \mathsf{one} = \mathsf{one} \tag{3}$$

$$\mathsf{dec}(x_{\mathsf{sk}}, \mathsf{penc}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{rand}}, x_{\mathsf{plain}})) = x_{\mathsf{plain}} \tag{4}$$

$$\mathsf{dec}(\mathsf{partial}(x_{\mathsf{sk}}, \mathit{ciph}), \mathit{ciph}) = x_{\mathsf{plain}} \tag{5}$$
where $\mathit{ciph} = \mathsf{penc}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{rand}}, x_{\mathsf{plain}})$

$$\mathsf{penc}(x_{\mathsf{pk}}, y_{\mathsf{rand}}, y_{\mathsf{plain}}) * \mathsf{penc}(x_{\mathsf{pk}}, z_{\mathsf{rand}}, z_{\mathsf{plain}}) = \mathsf{penc}(x_{\mathsf{pk}}, y_{\mathsf{rand}} \circ z_{\mathsf{rand}}, y_{\mathsf{plain}} + z_{\mathsf{plain}}) \tag{6}$$

$$\mathsf{checkspk}(x_{\mathsf{pk}}, \mathit{ball}, \mathsf{spk}(x_{\mathsf{pk}}, x_{\mathsf{rand}}, \mathsf{zero}, \mathit{ball})) = \mathsf{ok} \tag{7}$$
where $\mathit{ball} = \mathsf{penc}(x_{\mathsf{pk}}, x_{\mathsf{rand}}, \mathsf{zero})$

$$\mathsf{checkspk}(x_{\mathsf{pk}}, \mathit{ball}, \mathsf{spk}(x_{\mathsf{pk}}, x_{\mathsf{rand}}, \mathsf{one}, \mathit{ball})) = \mathsf{ok} \tag{8}$$
where $\mathit{ball} = \mathsf{penc}(x_{\mathsf{pk}}, x_{\mathsf{rand}}, \mathsf{one})$

Equation 5 allows plaintext M to be recovered from ciphertext $\mathsf{penc}(\mathsf{pk}(L), N, M)$ given partial decryption $\mathsf{partial}(L, \mathsf{penc}(\mathsf{pk}(L), N, M))$, when the partial decryption is constructed using the private key L. Equation 6 represents the homomorphic combination of ciphertexts. The equations 7 and 8 allow the verification of signatures of knowledge $\mathsf{spk}(T, N, M, \mathsf{penc}(T, N, M))$, when $M \in \{\mathsf{zero}, \mathsf{one}\}$. The remaining equations are standard.

Example 1 *Given randomness N, N', plaintexts $(M, M') \in \{(\mathsf{zero}, \mathsf{zero}), (\mathsf{zero}, \mathsf{one}), (\mathsf{one}, \mathsf{zero})\}$, and public key T, one can construct a signature of knowledge $L = \mathsf{spk}(T, N \circ N', M + M', \mathsf{penc}(T, N, M) * \mathsf{penc}(T, N', M'))$. Then* checkspk *applied to the public key T, the homomorphically combined ciphertexts $\mathsf{penc}(T, N, M) * \mathsf{penc}(T, N', M')$, and the signature L is equal to* ok *using Equations 3,6,7, and 8*

3.2.2 Helios process specification

In an election with two candidates, the trusted components are modelled by the administration process A_n^{ϕ} and voting process V defined in Figure 4. For

Figure 4 Helios process specification

Given the number of voters $n \geq 2$ and Helios process specification ϕ, the administration process A_n^ϕ and voting process V are defined below

$$
\begin{array}{lcl}
V & = & \nu\, r\,. \\
 & & \mathsf{let}\ ciph = \mathsf{penc}(z_{\mathsf{pk}}, r, x_{\mathsf{vote}})\ \mathsf{in} \\
 & & \mathsf{let}\ spk = \mathsf{spk}(z_{\mathsf{pk}}, r, x_{\mathsf{vote}}, ciph))\ \mathsf{in} \\
 & & \nu\, r'\,. \\
 & & \mathsf{let}\ ciph' = \mathsf{penc}(z_{\mathsf{pk}}, r', x'_{\mathsf{vote}})\ \mathsf{in} \\
 & & \mathsf{let}\ spk' = \mathsf{spk}(z_{\mathsf{pk}}, r', x'_{\mathsf{vote}}, ciph'))\ \mathsf{in} \\
 & & \mathsf{let}\ \widehat{spk} = \mathsf{spk}(z_{\mathsf{pk}}, r \circ r', x_{\mathsf{vote}} + x'_{\mathsf{vote}}, ciph * ciph')\ \mathsf{in} \\
 & & \overline{x_{\mathsf{auth}}}\langle (\!| ciph, ciph', spk, spk', \widehat{spk} |\!)\rangle \\
 & & \\
A_n^\phi & = & \nu\, sk_T, a_1, a_2, d\,. \ (\_ \mid BB_n^\phi \mid T \mid \{^{\mathsf{pk}(sk_T)}/_{z_{\mathsf{pk}}}\}) \\
 & & \\
BB_n^\phi & = & a_1(y_1)\,.\ \overline{c}\langle y_1\rangle\,.\ a_2(y_2)\,.\ \overline{c}\langle y_2\rangle\,. \\
 & & a_3(y_3)\,.\ \mathsf{if}\ \phi\{^{y_3}/_{y_{\mathsf{ballot}}}\}\ \mathsf{then} \\
 & & \cdots\ a_n(y_n)\,.\ \mathsf{if}\ \phi\{^{y_n}/_{y_{\mathsf{ballot}}}\}\ \mathsf{then} \\
 & & \mathsf{let}\ tally = \pi_1(y_1) * \cdots * \pi_1(y_n)\ \mathsf{in} \\
 & & \mathsf{let}\ tally' = \pi_2(y_1) * \cdots * \pi_2(y_n)\ \mathsf{in} \\
 & & \overline{d}\langle (\!| tally, tally' |\!)\rangle\,. \\
 & & d(y_{\mathsf{partial}})\,. \\
 & & \overline{c}\langle y_{\mathsf{partial}}\rangle\,. \\
 & & \overline{c}\langle (\!| \mathsf{dec}(\pi_1(y_{\mathsf{partial}}), tally), \mathsf{dec}(\pi_2(y_{\mathsf{partial}}), tally') |\!)\rangle \\
 & & \\
T & = & d(y_{\mathsf{tally}})\,. \\
 & & \overline{d}\langle (\!| \mathsf{partial}(sk_T, \pi_1(y_{\mathsf{tally}})), \mathsf{partial}(sk_T, \pi_2(y_{\mathsf{tally}})) |\!)\rangle
\end{array}
$$

generality, the administration process A_n^ϕ is parametrised by the number of voters n and a formula ϕ; the latter corresponds to the checks performed by the bulletin board before accepting a ballot.

Definition 3 (Helios process specification) *A formula ϕ is a* Helios process specification*, if* $\mathrm{fv}(\phi) \subseteq \{y_1, y_2, y_{\mathsf{ballot}}, z_{\mathsf{pk}}\}$.

The voting process V contains free variables $x_{\mathsf{vote}}, x'_{\mathsf{vote}}$ to represent the voter's vote (which is expected to be encoded as constants zero and one), and the free variable x_{auth} represents the channel shared by the voter and the bulletin board. The definition of the process V corresponds to the description of the browser script (Figure 1). The administration process A_n^ϕ is parametrised by the number of voters n and Helios process specification ϕ. The restricted name sk_T models the tallier's secret key and the public part $\mathsf{pk}(sk_T)$ is included in the process's frame. The restricted names a_1, a_2 model authentic channels between the two honest voters and the bulletin board, and the channel name d

captures the authentic channel with the honest trustee. To ensure the adversary has access to messages sent on private channels, communication is relayed on the public channel c. The sub-process BB_n^ϕ represents the bulletin board and T represents the tallier. The bulletin board accepts ballots from each voter and checks they are valid using the Helios process specification ϕ (this predicate will be discussed in more detail below). Once all ballots have been submitted, the bulletin board homomorphically combines the ciphertexts and sends the encrypted tallies to the tallier for decryption. (The necessity for all voters to participate is included for simplicity, in particular, our bulletin does not weed ballots containing invalid proofs.) The tallier receives the homomorphic combinations of ballots y_{tally} and derives a partial decryption for each candidate; these partial decryptions are sent to the bulletin board and the election result is published.

Example 2 *Given a Helios process specification ϕ, an election with voters $\mathcal{A}$ and $\mathcal{B}$ who select votes $(m_1, m_1'), (m_2, m_2') \in \{(\mathsf{zero}, \mathsf{zero}), (\mathsf{zero}, \mathsf{one}), (\mathsf{one}, \mathsf{zero})\}$ and such that the other $n-2$ voters are controlled by the adversary, can be modelled by the process $A_n^\phi[V\{{}^{a_1}/_{x_{\mathsf{auth}}}\}\sigma \mid V\{{}^{a_2}/_{x_{\mathsf{auth}}}\}\tau]$, where $\sigma = \{{}^{m_1}/_{x_{\mathsf{vote}}}, {}^{m_1'}/_{x_{\mathsf{vote}}'}\}$ and $\tau = \{{}^{m_2}/_{x_{\mathsf{vote}}}, {}^{m_2'}/_{x_{\mathsf{vote}}'}\}$.*

Ballot validity. In Helios 2.0, the election officer considers a ballot to be valid if the signature proofs of knowledge hold. Accordingly, we can model the Helios administration by the process $A_n^{\phi_{\mathsf{orig}}}$ where the Helios process specification ϕ_{orig} is defined as follows.

$$\begin{aligned}\phi_{\mathsf{orig}} \triangleq\ & \mathsf{checkspk}(z_{\mathsf{pk}}, \pi_1(y_{\mathsf{ballot}}), \pi_3(y_{\mathsf{ballot}})) = \mathsf{ok} \\ & \wedge\ \mathsf{checkspk}(z_{\mathsf{pk}}, \pi_2(y_{\mathsf{ballot}}), \pi_4(y_{\mathsf{ballot}})) = \mathsf{ok} \\ & \wedge\ \mathsf{checkspk}(z_{\mathsf{pk}}, \pi_1(y_{\mathsf{ballot}}) * \pi_2(y_{\mathsf{ballot}}), \pi_5(y_{\mathsf{ballot}})) = \mathsf{ok}\end{aligned}$$

3.2.3 Limitations

The limitations of our model, which we introduced to simplify the presentation, are detailed below. We consider a model with with two honest voters for compatibility with the (standard) definitions of ballot secrecy [4, 5, 32]. The administrative process A_n^ϕ enforces an ordering on voters (namely, the voter using private channel a_1 must vote first, followed by the voter using private channel a_2, and then any remaining voters – controlled by the adversarial environment – can vote) and A_n^ϕ does not permit revoting. The signature and equational theory do not capture low-level technical details surrounding the correct construction of public keys; in particular, we do not use signatures of knowledge to verify correct key construction. We also omit signatures of knowledge that demonstrate correct construction of partial decryptions.

4 Summary and future work

In this report we present a cryptographic description of the Helios protocol and a model in the applied pi calculus. Our formal description can be used to analyse ballot secrecy based upon Kremer *et al.* [4, 5, 32]. In this definition, two voters $\mathcal{A}$, $\mathcal{B}$ and two candidates t, t' are considered. Ballot secrecy is captured by the assertion that an adversary (controlling arbitrary many dishonest voters) cannot distinguish between a situation in which voter $\mathcal{A}$ votes for candidate t and voter $\mathcal{B}$ votes for candidate t', from another situation in which $\mathcal{A}$ votes t' and $\mathcal{B}$ votes t. This can be expressed by the following equivalence.

$$\mathcal{A}(t) \mid \mathcal{B}(t') \approx_l \mathcal{A}(t') \mid \mathcal{B}(t)$$

This formal definition of ballot secrecy has been used to analyse the electronic voting protocols due to: Fujioka, Okamoto & Ohta [1], Okamoto [2], Lee *et al.* [34], and Juels, Catalano & Jakobsson [3, 35, 36]. It therefore seems natural to check whether Helios satisfies ballot secrecy as well, and formally this can be expressd as follows.

Definition 4 (Ballot secrecy) *Given a Helios process specification ϕ, we say* ballot secrecy is satisfied *if for all $(m_1, m_1'), (m_2, m_2') \in \{(\mathsf{zero}, \mathsf{zero}), (\mathsf{zero}, \mathsf{one}), (\mathsf{one}, \mathsf{zero})\}$ and integers $n \geq 2$, we have*

$$A_n^\phi[V\{^{a_1}/_{x_{\mathsf{auth}}}\}\sigma \mid V\{^{a_2}/_{x_{\mathsf{auth}}}\}\tau] \approx A_n^\phi[V\{^{a_1}/_{x_{\mathsf{auth}}}\}\tau \mid V\{^{a_2}/_{x_{\mathsf{auth}}}\}\sigma]$$

where $\sigma = \{^{m_1}/_{x_{\mathsf{vote}}}, ^{m_1'}/_{x_{\mathsf{vote}}'}\}$ and $\tau = \{^{m_2}/_{x_{\mathsf{vote}}}, ^{m_2'}/_{x_{\mathsf{vote}}'}\}$.

Future work will analyse whether the Helios process specification ϕ_{orig} satisfies ballot secrecy.

Appendix

A Signatures of knowledge

Helios is reliant on signatures of knowledge to ensure secrecy and integrity of the ElGamal scheme, and to ensure voters encrypt valid votes. This appendix presents suitable cryptographic primitives. Let $\mathcal{H}$ denote a hash function. In Helios, $\mathcal{H}$ is defined to be SHA-256.

Knowledge of discrete logs. Given the aforementioned cryptographic parameters (p, q, g), a signature of knowledge demonstrating knowledge of a discrete logarithm $h = \log_g g^x$ can be derived, and verified, as defined by [37, 38, 39].

Sign. Given x, select a random nonce $w \in_R \mathbb{Z}_q^*$. Compute witness $g' = g^w \bmod p$, challenge $c = \mathcal{H}(g') \bmod q$ and response $s = w + c \cdot x \bmod q$.

Verify. Given h and signature g', s, check $g^s \equiv g' \cdot h^c \pmod{p}$, where $c = \mathcal{H}(g') \bmod q$.

A valid proof asserts knowledge of x such that $x = \log_g h$; that is, $h \equiv g^x \bmod p$.

Equality between discrete logs. Given the aforementioned cryptographic parameters (p, q, g), a signature of knowledge demonstrating equality between discrete logarithms $\log_f f^x$ and $\log_g g^x$ can be derived, and verified, as defined by [21, 22].

Sign. Given f, g, x, select a random nonce $w \in_R \mathbb{Z}_q^*$. Compute witnesses $f' = f^w \bmod p$ and $g' = g^w \bmod p$, challenge $c = \mathcal{H}(f', g') \bmod q$ and response $s = w + c \cdot x \bmod q$.

Verify. Given f, g, h, k and signature f', g', s, check $f^s \equiv f' \cdot h^c \pmod{p}$ and $g^s \equiv g' \cdot k^c \pmod{p}$, where $c = \mathcal{H}(f', g') \bmod q$.

A valid proof asserts $\log_f h = \log_g k$; that is, there exists x, such that $h \equiv f^x \bmod p$ and $k \equiv g^x \bmod p$. This signature of knowledge scheme can be extended to a disjunctive proof of equality between discrete logs (see below).

For our purposes, given a ciphertext (a, b), each trustee would derive a signature on g, a, x_i, where x_i is the trustee's private key share. The ith trustee's signature g_i', a_i', c_i, s_i would be verified with respect to g, a, h_i, k_i, where h_i is the trustee's share of the public key and k_i is the trustee's partial decryption; that is, the proof asserts $\log_g h_i = \log_a k_i$, as required for integrity of decryption.

Disjunctive proof of equality between discrete logs. Given the aforementioned cryptographic parameters (p, q, g), a signature of knowledge demonstrating that a ciphertext (a, b) contains either 0 or 1 (without revealing which), can be constructed by proving that either $\log_g a = \log_h b$ or $\log_g a = \log_h b/g^m$; that is, a signature of knowledge demonstrating a disjunct proof of equality between discrete logarithms [18, 20]. Observe for a valid ciphertext (a, b) that $a \equiv g^r \bmod p$ and $b \equiv h^r \cdot g^m \bmod p$ for some nonce $r \in \mathbb{Z}_q^*$; hence the former disjunct $\log_g g^r = \log_h h^r \cdot g^m$ is satisfied when $m = 0$, and the latter $\log_g g^r = \log_h (h^r \cdot g^m)/g^m$ when $m = 1$.

This technique is generalised by [12] to allow a signature of knowledge demonstrating that a ciphertext (a, b) contains message m, where $m \in \{\mathsf{min}, \ldots, \mathsf{max}\}$ for some system parameters $\mathsf{min}, \mathsf{max} \in \mathbb{N}$. Formally, a signature of knowledge demonstrating a disjunct proof of equality between discrete logarithms can be derived, and verified, as follows [12, 18, 20].

Sign. Given ciphertext (a, b) such that $a \equiv g^r \bmod p$ and $b \equiv h^r \cdot g^m \bmod p$ for some nonce $r \in \mathbb{Z}_q^*$, where plaintext $m \in \{\mathsf{min}, \ldots, \mathsf{max}\}$. For all $i \in \{\mathsf{min}, \ldots, m-1, m+1, \ldots, \mathsf{max}\}$, compute challenge $c_i \in_R \mathbb{Z}_q^*$, response $s_i \in_R \mathbb{Z}_q^*$ and witnesses $a_i = g^{s_i}/a^{c_i} \bmod p$ and $b_i = h^{s_i}/(b/g^i)^{c_i} \bmod p$. Select a random nonce $w \in_R \mathbb{Z}_q^*$. Compute witnesses $a_m = g^w \bmod p$ and $b_m = h^w \bmod p$, challenge $c_m = \mathcal{H}(a_{\mathsf{min}}, b_{\mathsf{min}}, \ldots, a_{\mathsf{max}}, b_{\mathsf{max}}) - \sum_{i \in \{\mathsf{min}, \ldots, m-1, m+1, \ldots, \mathsf{max}\}} c_i \pmod{q}$ and response $s_m = w + r \cdot c_m \bmod q$.

Verify. Given (a, b) and $(a_{\mathsf{min}}, b_{\mathsf{min}}, c_{\mathsf{min}}, s_{\mathsf{min}}, \ldots, a_{\mathsf{max}}, b_{\mathsf{max}}, c_{\mathsf{max}}, s_{\mathsf{max}})$, for each $\mathsf{min} \leq i \leq \mathsf{max}$ check $g^{s_i} \equiv a_i \cdot a^{c_i} \pmod{p}$ and $h^{s_i} \equiv b_i \cdot (b/g^i)^{c_i} \pmod{p}$. Finally, check $\mathcal{H}(a_{\mathsf{min}}, b_{\mathsf{min}}, \ldots, a_{\mathsf{max}}, b_{\mathsf{max}}) \equiv \sum_{\mathsf{min} \leq i \leq \mathsf{max}} c_i \pmod{q}$.

A valid proof asserts that (a, b) is a ciphertext containing the message m such that $m \in \{\mathsf{min}, \ldots, \mathsf{max}\}$.

References

[1] Fujioka, A., Okamoto, T., Ohta, K.: A Practical Secret Voting Scheme for Large Scale Elections. In: AUSCRYPT'92: Workshop on the Theory and Application of Cryptographic Techniques. Volume 718 of LNCS., Springer (1992) 244–251

[2] Okamoto, T.: Receipt-Free Electronic Voting Schemes for Large Scale Elections. In: SP'97: 5th International Workshop on Security Protocols. Volume 1361 of LNCS., Springer (1998) 25–35

[3] Juels, A., Catalano, D., Jakobsson, M.: Coercion-Resistant Electronic Elections. Cryptology ePrint Archive, Report 2002/165 (2002)

[4] Kremer, S., Ryan, M.D.: Analysis of an Electronic Voting Protocol in the Applied Pi Calculus. In: ESOP'05: 14th European Symposium on Programming. Volume 3444 of LNCS., Springer (2005) 186–200

[5] Delaune, S., Kremer, S., Ryan, M.: Coercion-Resistance and Receipt-Freeness in Electronic Voting. In: CSFW'06: 19th Computer Security Foundations Workshop, IEEE Computer Society (2006) 28–42

[6] Backes, M., Hriţcu, C., Maffei, M.: Automated Verification of Remote Electronic Voting Protocols in the Applied Pi-calculus. In: CSF'08: 21st Computer Security Foundations Symposium, IEEE Computer Society (2008) 195–209

[7] Participants of the Dagstuhl Conference on Frontiers of E-Voting: Dagstuhl Accord. `http://www.dagstuhlaccord.org/` (2007)

[8] Kremer, S., Ryan, M.D., Smyth, B.: Election verifiability in electronic voting protocols. In: ESORICS'10: 15th European Symposium on Research in Computer Security. Volume 6345 of LNCS., Springer (2010) 389–404

[9] Chaum, D., Ryan, P.Y., Schneider, S.: A Practical Voter-Verifiable Election Scheme. In: ESORICS'05: 10th European Symposium On Research In Computer Security. Volume 3679 of LNCS., Springer (2005) 118–139

[10] Adida, B.: Advances in Cryptographic Voting Systems. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology (2006)

[11] Adida, B.: Helios: Web-based Open-Audit Voting. In: USENIX Security'08: 17th USENIX Security Symposium, USENIX Association (2008) 335–348

[12] Adida, B., Marneffe, O., Pereira, O., Quisquater, J.: Electing a University President Using Open-Audit Voting: Analysis of Real-World Use of Helios. In: EVT/WOTE'09: Electronic Voting Technology Workshop/Workshop on Trustworthy Elections, USENIX Association (2009)

[13] Smyth, B., Ryan, M.D., Kremer, S., Kourjieh, M.: Towards automatic analysis of election verifiability properties. In: ARSPA-WITS'10: Joint Workshop on Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security. Volume 6186 of LNCS., Springer (2010) 165–182 To appear.

[14] Smyth, B.: Formal verification of cryptographic protocols with automated reasoning. PhD thesis, School of Computer Science, University of Birmingham (2010)

[15] Benaloh, J., Vaudenay, S., Quisquater, J.: Final Report of IACR Electronic Voting Committee. International Association for Cryptologic Research. `http://www.iacr.org/elections/eVoting/finalReportHelios_2010-09-27.html` (Sept 2010)

[16] Haber, S., Benaloh, J., Halevi, S.: The Helios e-Voting Demo for the IACR. International Association for Cryptologic Research. `http://www.iacr.org/elections/eVoting/heliosDemo.pdf` (May 2010)

[17] University, P.: Princeton election server. `https://princeton-helios.appspot.com/` (2010)

[18] Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In: CRYPTO'94: 14th International Cryptology Conference. Volume 839 of LNCS., Springer (1994) 174–187

[19] Cramer, R., Gennaro, R., Schoenmakers, B.: A Secure and Optimally Efficient Multi-Authority Election Scheme. In: EUROCRYPT'97: 16th International Conference on the Theory and Applications of Cryptographic Techniques. Volume 1233 of LNCS., Springer (1997) 103–118

[20] Schoenmakers, B.: Voting Schemes. In Atallah, M.J., Blanton, M., eds.: Algorithms and Theory of Computation Handbook, Second Edition, Volume 2: Special Topics and Techniques. CRC Press (2009)

[21] Pedersen, T.P.: A Threshold Cryptosystem without a Trusted Party. In: EUROCRYPT'91: 10th International Conference on the Theory and Applications of Cryptographic Techniques. Number 547 in LNCS, Springer (1991) 522–526

[22] Chaum, D., Pedersen, T.P.: Wallet Databases with Observers. In: CRYPTO'92: 12th International Cryptology Conference. Volume 740 of LNCS., Springer (1993) 89–105

[23] ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory **31**(4) (1985) 469–472

[24] Shanks, D.: Class number, a theory of factorization and genera. In: Number Theory Institute. Volume 20 of Symposia in Pure Mathematics., American Mathematical Society (1971) 415–440

[25] Lenstra, A.K., Lenstra Jr., H.W.: Algorithms in Number Theory. In Leeuwen, J., ed.: Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity. MIT Press (1990) 673–716

[26] Rudolph, C.: Covert Identity Information in Direct Anonymous Attestation (DAA). In: SEC'07: 22nd International Information Security Conference. Volume 232 of International Federation for Information Processing (IFIP)., Springer (2007) 443–448

[27] Leung, A., Chen, L., Mitchell, C.J.: On a Possible Privacy Flaw in Direct Anonymous Attestation (DAA). In: Trust'08: 1st International Conference on Trusted Computing and Trust in Information Technologies. Number 4968 in LNCS, Springer (2008) 179–190

[28] Benaloh, J.: Simple Verifiable Elections. In: EVT'06: Electronic Voting Technology Workshop, USENIX Association (2006)

[29] Benaloh, J.: Ballot Casting Assurance via Voter-Initiated Poll Station Auditing. In: EVT'07: Electronic Voting Technology Workshop, USENIX Association (2007)

[30] Abadi, M., Fournet, C.: Mobile values, new names, and secure communication. In: POPL'01: 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, ACM Press (2001) 104–115

[31] Ryan, M.D., Smyth, B.: Applied pi calculus. In Cortier, V., Kremer, S., eds.: Formal Models and Techniques for Analyzing Security Protocols. IOS Press (2011) To appear.

[32] Delaune, S., Kremer, S., Ryan, M.D.: Verifying privacy-type properties of electronic voting protocols. Journal of Computer Security **17**(4) (July 2009) 435–487

[33] Pereira, O., Adida, B., Marneffe, O.: Bringing open audit elections into practice: Real world uses of helios. Swiss e-voting workshop, `https://www.e-voting-cc.ch/images/sevot10/slides/helios_20100906.pdf`. See also `http://www.uclouvain.be/crypto/electionmonitor/` (2010)

[34] Lee, B., Boyd, C., Dawson, E., Kim, K., Yang, J., Yoo, S.: Providing Receipt-Freeness in Mixnet-Based Voting Protocols. In: ICISC'03: 6th International Conference on Information Security and Cryptology. Volume 2971 of LNCS., Springer (2004) 245–258

[35] Juels, A., Catalano, D., Jakobsson, M.: Coercion-Resistant Electronic Elections. In: WPES'05: 4th Workshop on Privacy in the Electronic Society, ACM Press (2005) 61–70 See also `http://www.rsa.com/rsalabs/node.asp?id=2860`.

[36] Juels, A., Catalano, D., Jakobsson, M.: Coercion-Resistant Electronic Elections. In Chaum, D., Jakobsson, M., Rivest, R.L., Ryan, P.Y., eds.: Towards Trustworthy Elections: New Directions in Electronic Voting. Volume 6000 of LNCS. Springer (2010) 37–63

[37] Chaum, D., Evertse, J., Graaf, J., Peralta, R.: Demonstrating Possession of a Discrete Logarithm Without Revealing It. In: CRYPTO'86: 6th International Cryptology Conference. Volume 263 of LNCS., Springer (1987) 200–212

[38] Chaum, D., Evertse, J., Graaf, J.: An Improved Protocol for Demonstrating Possession of Discrete Logarithms and Some Generalizations. In: EUROCRYPT'87: 4th International Conference on the Theory and Applications of Cryptographic Techniques. Volume 304 of LNCS., Springer (1988) 127–141

[39] Schnorr, C.P.: Efficient Identification and Signatures for Smart Cards. In: CRYPTO'89: 9th International Cryptology Conference. Volume 435 of LNCS., Springer (1990) 239–252