Practical everlasting privacy

Myrto Arapinis[1], Véronique Cortier[2], Steve Kremer[2], and Mark Ryan[1]

[1] School of Computer Science, University of Birmingham
[2] LORIA, CNRS, France

Abstract. Will my vote remain secret in 20 years? This is a natural question in the context of electronic voting, where encrypted votes may be published on a bulletin board for verifiability purposes, but the strength of the encryption is eroded with the passage of time. The question has been addressed through a property referred to as *everlasting privacy*. Perfect everlasting privacy may be difficult or even impossible to achieve, in particular in remote electronic elections. In this paper, we propose a definition of *practical everlasting privacy*. The key idea is that in the future, an attacker will be more powerful in terms of computation (he may be able to break the cryptography) but less powerful in terms of the data he can operate on (transactions between a vote client and the vote server may not have been stored).
We formalize our definition of everlasting privacy in the applied-pi calculus. We provide the means to characterize what an attacker can break in the future in several cases. In particular, we model this for perfectly hiding and computationally binding primitives (or the converse), such as Pedersen commitments, and for symmetric and asymmetric encryption primitives. We adapt existing tools, in order to allow us to automatically prove everlasting privacy. As an illustration, we show that several variants of Helios (including Helios with Pedersen commitments) and a protocol by Moran and Naor achieve practical everlasting privacy, using the ProVerif and the AKiSs tools.

1 Introduction

Electronic voting schemes such as Helios [2], JCJ/Civitas [14, 8], and Prêt-à-Voter [7] aim simultaneously to guarantee *vote privacy* (that is, the link between the voter and her vote will not be revealed), and *outcome verifiability* (that is, voters and observers can check that the declared outcome is correct). A common way to achieve verifiability is to publish a "bulletin board" that contains all encrypted votes (indeed, this is the way it is done in the systems cited above). The strength and key-length of the encryption should be chosen so that decryption by an attacker is impossible for as long as the votes are expected to remain private. To prevent coercer reprisal not just to the voter but also to her descendants, one may want vote privacy for up to 100 years.

Unfortunately, however, it is not possible to predict in any reliable way how long present-day encryptions will last. Weaknesses may be found in encryption algorithms, and computers will certainly continue to get faster. A coercer can plausibly assert that a voter should follow the coercer's wishes because the bulletin board will reveal in (say) 10 years whether the voter followed the coercer's instructions. For this reason, systems with "everlasting privacy" have been introduced by [18]. These systems do not rely on encryptions whose strength may be eroded, but on commitments that are *perfectly* or *information-theoretically hiding*. These systems have computational verifiability instead of perfect verifiability, and are considered less usable and computationally more expensive than systems relying on encryptions. More recently, schemes have been proposed with a weaker form of everlasting privacy (e.g., [10, 12]); they rely on encryptions for counting votes, but use commitments rather than encryptions for verifiability purposes. Thus, the bulletin board which only publishes the commitments does not weaken the privacy provided by the underlying scheme. Although the encrypted votes must be sent to the election administrators, it is assumed that these communications cannot be intercepted and stored *en masse*. We call this weaker form of everlasting privacy *practical everlasting privacy*.

Symbolic models for security protocol analysis have been used to model both privacy properties (e.g., [11, 3, 13]) and verifiability properties (e.g.,[16, 17]) of voting systems, but they are currently not capable of distinguishing *perfect* versus *computational* notions of privacy, or indeed, of verifiability. Our aim in this paper is to extend the model to allow these distinctions. We focus on practical everlasting privacy, and use our definitions to verify whether particular schemes satisfy that property.

Our contributions. Our first and main contribution is a general and practical definition of everlasting privacy. The key idea is that, in the future, an attacker will be more powerful in terms of computation (he may be able to break cryptography) but less powerful in terms of the data he can operate on (transactions between a vote client and the vote server may not have been stored). We therefore distinguish between standard communication channels (on which eavesdropping may be possible, but requires considerable effort) and *everlasting channels*, on which the information is intentionally published and recorded permanently (e.g. web pages that serve as a public bulletin board). Formally, we model everlasting privacy in the applied-pi calculus [1], a framework well-adapted to security protocols and already used to define privacy [11] and verifiability [16]. Our definitions apply not only to voting protocols but also to situations where forward secrecy is desirable, such as for instance untraceability in RFID protocols.

Modeling everlasting privacy also requires to precisely model what an attacker can break in the future. Our second contribution is a characterization, for several primitives, of what can be broken. The first natural primitive is encryption, for which we provide an equational theory that models the fact that private keys can be retrieved from public keys, or even from ciphertexts. Some other primitives have been primarily designed to achieve everlasting privacy. This is the case of *perfectly hiding* and *computationally binding* primitives, such as Pedersen commitments [19]. Intuitively, perfectly hiding means that the hidden secret cannot be retrieved even by a computationally unbounded adversary, while computationally binding means that, binding is ensured only for a (polynomially) limited attacker. We provide an equational theory that models such perfectly hiding and computationally binding primitives in general.

As an application, we study everlasting privacy for several variants of Helios [2], an e-voting protocol used for electing the president of the University of Louvain and board members of the IACR[3]. We study in particular its latest variants with Pedersen

[3] International Association for Cryptologic Research

commitments [12], designed to achieve everlasting privacy, still providing full verifiability. We also model and prove everlasting privacy of a (simplified) version of Moran and Naor's protocol [18]. Interestingly, we were able to adapt algorithms in existing tools to automate the verification of everlasting privacy and we use adapted versions of the AKisS [6] and ProVerif [4] tools to analyze everlasting privacy for half a dozen of protocols.

Outline. In the following section we recall the applied pi calculus and introduce notations and basic definitions. In Section 3 we define new equivalence relations, namely forward and everlasting indistinguishability. Then, in Section 4 we instantiate these equivalences to the case of voting protocols, define everlasting privacy and illustrate this property on several examples. In Section 5 we present a modeling of perfectly hiding and computationally binding (and vice-versa) primitives in the applied pi calculus. In particular we model Pedersen commitments, which are for studying two protocols that provide everlasting privacy. In Section 6 we discuss tool support for automatically proving everlasting indistinguishability before concluding.

2 The applied pi calculus

The applied pi calculus [1] is a language for modeling distributed systems and their interactions. It extends the pi calculus with an equational theory, which is particularly useful for modeling cryptographic protocols. The following subsections describe the syntax and semantics of the calculus.

2.1 Syntax

Terms. The calculus assumes an infinite set of names $\mathcal{N} = \{a, b, c, \ldots\}$, an infinite set of variables $\mathcal{V} = \{x, y, z, \ldots\}$ and a finite signature Σ, that is, a finite set of function symbols each with an associated arity. We use meta-variables u, v, w to range over both names and variables. Terms $M, N, T, \ldots$ are built by applying function symbols to names, variables and other terms. Tuples $M_1, \ldots, M_l$ are occasionally abbreviated $\tilde{M}$. We write $\{^{M_1}/_{u_1}, \ldots, ^{M_l}/_{u_l}\}$ for substitutions that replace $u_1, \ldots, u_l$ with $M_1, \ldots, M_l$. The applied pi calculus relies on a simple type system. Terms can be of sort $\mathsf{Channel}$ 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 signature Σ is equipped with an equational theory E, that is a finite set of equations of the form $M = N$. We define $=_\mathsf{E}$ as the smallest equivalence relation on terms, that contains E and is closed under application of function symbols, substitution of terms for variables and bijective renaming of names.

Example 1. A standard signature for pairing and encryption is:

$$\Sigma_\mathsf{enc} = \{0, 1, \langle \_, \_ \rangle, \mathsf{fst}(\_), \mathsf{snd}(\_), \mathsf{pk}(\_), \mathsf{aenc}(\_, \_, \_), \mathsf{adec}(\_, \_), \mathsf{senc}(\_, \_, \_), \mathsf{sdec}(\_, \_)\}$$

The term $\langle m_1, m_2 \rangle$ represents the concatenation of m_1 and m_2, with associated projectors $\mathsf{fst}(\_)$ and $\mathsf{snd}(\_)$. The term $\mathsf{aenc}(k, r, m)$ represents the asymmetric encryption of

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

where u is either a name or variable of channel sort.

Fig. 1. Applied pi calculus grammar

message m with public key k and randomness r while the associated decryption operator is adec. Similarly, $\mathsf{senc}(k, r, m)$ represents the symmetric encryption of message m with key k and randomness r. The associated decryption operator is sdec. The properties of these primitives are modeled by the following standard equational theory $\mathsf{E}_{\mathsf{enc}}$:

$$\mathsf{E}_{\mathsf{enc}} = \left\{ \begin{array}{r} \mathsf{fst}(\langle x, y\rangle) = x \\ \mathsf{snd}(\langle x, y\rangle) = y \\ \mathsf{adec}(x, \mathsf{aenc}(\mathsf{pk}(x), y, z)) = z \\ \mathsf{sdec}(x, \mathsf{senc}(x, y, z)) = z \end{array} \right\}$$

Processes. The grammar for processes is shown in Figure 1. Plain processes are standard. Extended processes introduce *active substitutions* which generalize the classical let construct: the process $\nu x.(\{^{M}/_{x}\} \mid P)$ corresponds exactly to the process $\text{let } x = M \text{ in } P$. As usual names and variables have scopes which are delimited by restrictions and by inputs. All substitutions are assumed to be cycle-free.

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

Example 2. Throughout the paper we illustrate our definitions with a simplified version of the Helios voting system [2]. Two techniques can be used for tallying in Helios: either a homomorphic tally based on El Gamal encryption, or a tally based on mixnets. We present here the version with mixnets.

1. The voter V computes her ballot by encrypting her vote with the public key $\mathsf{pk}(skE)$ of the election. The corresponding secret key is shared among several election authorities. Then she casts her ballot together with her identity on an authenticated channel. Upon receiving the ballot, the administrator simply publishes it on a public web page (after having checked that V is entitled to vote).
2. Once the voting phase is over, the votes are shuffled and reencrypted through mixnets. The permuted and rerandomized votes are again published on the public web page (together with a zero knowledge proof of correct reencryption and mixing).
3. Finally, the authorities decrypt the rerandomized votes and the administrator publishes the decrypted votes (with a zero knowledge proof of correct decryption).

The process representing the voter is parametrized by her vote v, and her identity id.

$$V(auth, id, v) \overset{\mathsf{def}}{=} \nu r. \overline{auth}\langle\langle id, \mathsf{aenc}(\mathsf{pk}(skE), r, v)\rangle\rangle$$

The administrator BB receives votes on private authenticated channels and publishes the votes. It is parametrized by the authenticated channels of the voters. Then the ballots are forwarded to the tally T over the private channel c. The tally consists in decrypting the vote. The shuffle through mixnets is modeled simply, by non deterministic parallel composition after all ballots have been received. For simplicity, we consider here an election system for three voters.

$$\begin{aligned}
&BB(a_1, a_2, a_3) \overset{\mathsf{def}}{=} \nu c.\ a_1(x).\ \overline{bb}\langle x\rangle.\ \overline{c}\langle x\rangle \mid a_2(y).\ \overline{bb}\langle y\rangle.\ \overline{c}\langle y\rangle \mid a_3(z).\ \overline{bb}\langle z\rangle.\ \overline{c}\langle z\rangle \mid T\\
&T \overset{\mathsf{def}}{=} c(x').c(y').c(z').\\
&\quad (\overline{bb}\langle \mathsf{adec}(skE, \mathsf{snd}(x'))\rangle \mid \overline{bb}\langle \mathsf{adec}(skE, \mathsf{snd}(y'))\rangle \mid \overline{bb}\langle \mathsf{adec}(skE, \mathsf{snd}(z'))\rangle)
\end{aligned}$$

The process H then represents the whole Helios system with two honest voters and one dishonest voter (which does therefore not need to be explicitly specified and whose authenticated channel is public).

$$\begin{aligned}
&H \overset{\mathsf{def}}{=} \nu skE.\ \nu auth_1.\ \nu auth_2.\\
&\overline{bb}\langle \mathsf{pk}(skE)\rangle.\ (V(auth_1, id_1, a) \mid V(auth_2, id_2, b) \mid BB(auth_1, auth_2, auth_3))
\end{aligned}$$

The first honest voter casts the vote a while the second honest voter casts the vote b.

2.2 Semantics

The operational semantics of the applied pi calculus is defined by the means of two relations: structural equivalence and internal reductions. *Structural equivalence* ($\equiv$) is the smallest equivalence relation closed under α-conversion of both bound names and variables and application of evaluation contexts such that:

$$\begin{array}{rlrl}
A \mid 0 &\equiv A & \nu n.0 &\equiv 0\\
A \mid (B \mid C) &\equiv (A \mid B) \mid C & \nu u.\nu w.A &\equiv \nu w.\nu u.A\\
A \mid B &\equiv B \mid A & A \mid \nu u.B &\equiv \nu u.(A \mid B)\\
!P &\equiv P \mid !P & & \text{if } u \notin \mathrm{fn}(A) \cup \mathrm{fv}(A)\\
\nu x.\{^M/_x\} &\equiv 0 & \{^M/_x\} &\equiv \{^N/_x\}\\
\{^M/_x\} \mid A &\equiv \{^M/_x\} \mid A\{^M/_x\} & & \text{if } M =_{\mathsf{E}} N
\end{array}$$

$$a(x).P \xrightarrow{a(M)} P\{^{M}/_{x}\} \qquad \frac{A \xrightarrow{\alpha} A' \qquad u \text{ does not occur in } \alpha}{\nu u.A \xrightarrow{\alpha} \nu u.A'}$$

$$\overline{a}\langle u\rangle.P \xrightarrow{\overline{a}\langle u\rangle} P \qquad \frac{A \xrightarrow{\alpha} A' \quad \mathrm{bv}(\alpha) \cap \mathrm{fv}(B) = \mathrm{bn}(\alpha) \cap \mathrm{fn}(B) = \emptyset}{A \mid B \xrightarrow{\alpha} A' \mid B}$$

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

Fig. 2. Labelled reductions.

Internal reduction ($\rightarrow$) is the smallest relation closed under structural equivalence, application of evaluation contexts satisfying the following rules:

$$\begin{array}{ll} \text{COMM} & \overline{c}\langle x\rangle.P \mid c(x).Q \rightarrow P \mid Q \\ \text{THEN} & \text{if } N = N \text{ then } P \text{ else } Q \rightarrow P \\ \text{ELSE} & \text{if } L = M \text{ then } P \text{ else } Q \rightarrow Q \\ & \quad \text{for ground terms } L, M \text{ where } L \neq_{\mathsf{E}} M \end{array}$$

Labelled reduction ($\xrightarrow{\alpha}$) extends the internal reduction and enables the environment to interact with the processes as defined in Figure 2. The label α is either an input, or the output of a channel name or a variable of base type.

We write $\Rightarrow$ for an arbitrary (possibly zero) number of internal reductions and $\overset{\alpha}{\Rightarrow}$ for $\Rightarrow\xrightarrow{\alpha}\Rightarrow$. Whenever the equational theory is not clear from the context we annotate the above relations by the equational theory and write e.g. $\rightarrow_{\mathsf{E}}$.

A *trace* of a process is the sequence of actions (i.e. labels) together with the corresponding sent messages. Formally, the set of traces of a process A is defined as follows. Note that it depends on the underlying equational theory E.

$$\mathsf{trace}_E(A) = \{(\alpha_1 \cdot \alpha_2 \cdot \ldots \cdot \alpha_n, \varphi(B)) \mid A \overset{\alpha_1}{\Rightarrow}_{\mathsf{E}} A_1 \overset{\alpha_2}{\Rightarrow}_{\mathsf{E}} \cdots A_{n-1} \overset{\alpha_n}{\Rightarrow}_{\mathsf{E}} B\}$$

Example 3. Consider the process H representing the Helios protocol as defined in Example 2. A possible execution for H is:

$$H \xRightarrow{\nu xk.\ \overline{bb}\langle xk\rangle} H_1$$
$$\xRightarrow{\nu x.\ \overline{bb}\langle x\rangle} \xRightarrow{\nu y.\ \overline{bb}\langle y\rangle} \xRightarrow{auth_3(\langle id_3, x\rangle)} \xRightarrow{\nu z.\ \overline{bb}\langle z\rangle} \xRightarrow{\nu x'.\ \overline{bb}\langle x'\rangle} \xRightarrow{\nu y'.\ \overline{bb}\langle y'\rangle} \xRightarrow{\nu z'.\ \overline{bb}\langle z'\rangle} H_2$$

where H_1 and H_2 are defined below (we omit the other intermediate processes). Note that H_2 is simply an active substitution.

$$H_1 = \nu skE.\ \nu auth_1.\ \nu auth_2.\ \nu r_1.$$
$$(\{^{\mathsf{pk}(skE)}/_{xk}\} \mid V(auth_1, id_1, a) \mid V(auth_2, id_2, b) \mid\ BB(auth_1, auth_2, auth_3))$$

$$H_2 = \nu skE.\ \nu auth_1.\ \nu auth_2.\ \nu r_1.\ \nu r_2.\{^{\mathsf{pk}(skE)}/_{xk}\} \mid \{^{a}/_{x'}, {}^{b}/_{y'}, {}^{a}/_{z'}\} \mid$$
$$\{^{\langle id_1, \mathsf{aenc}(\mathsf{pk}(skE), r_1, a)\rangle}/_{x}, {}^{\langle id_2, \mathsf{aenc}(\mathsf{pk}(skE), r_2, b)\rangle}/_{y}, {}^{\langle id_3, \mathsf{aenc}(\mathsf{pk}(skE), r_1, a)\rangle}/_{z}\}$$

This execution trace corresponds to the case where the two honest voters cast their vote as expected, while the dishonest voter replays the first voter's ballot. As we shall see in Example 5, this corresponds to the attack on privacy discovered in [9].

2.3 Equivalence Relations for Processes

Privacy is often stated in terms of equivalence [11]. We recall here the definitions of static and trace equivalence.

Sequences of messages are often stored as *frames*. Formally, a frame is an extended process built from 0 and active substitutions $\{^{M}/_{x}\}$, and closed by parallel composition and restriction. The *domain* of a frame $\phi = \nu\tilde{n}.\ \{^{M_1}/_{x_1}, \ldots, {}^{M_n}/_{x_n}\}$ such that $x_i \notin \tilde{n}$ is $\mathrm{dom}(\phi) = \{x_1, \ldots, x_n\}$. Every extended process A can be mapped to a frame $\varphi(A)$ by replacing every plain process in A with 0. The frame $\varphi(A)$ represents the static knowledge output by a process to its environment.

Two frames are indistinguishable to an attacker if it is impossible to build a test that allows to differentiate between the two.

Definition 1 (Static equivalence). *Given an equational theory* E *two frames* ϕ *and* ψ *are are statically equivalent, denoted* $\phi \sim_{\mathsf{E}} \psi$*, if* $\mathrm{dom}(\phi) = \mathrm{dom}(\psi)$ *and there exist* $\tilde{n}, \sigma, \tau$ *such that* $\phi \equiv \nu\tilde{n}.\sigma$*,* $\psi \equiv \nu\tilde{n}.\tau$ *and for all terms* M, N *such that* $\tilde{n} \cap (\mathrm{fn}(M) \cup \mathrm{fn}(N)) = \emptyset$*, we have* $M\sigma =_{\mathsf{E}} N\sigma$ *if and only if* $M\tau =_{\mathsf{E}} N\tau$*. By abuse of notation, we may write* $M\phi$ *instead of* $M\sigma$ *when* σ *is clear from the context.*

Example 4. Let $\mathsf{E}_{\mathsf{enc}}$ be the equational theory defined at Example 1. Let H_2 be the process/frame defined in Example 3. Let $\phi = \varphi(H_2)$ ($= H_2$ actually). Consider the following frame ψ.

$$\psi = \nu skE.\ \nu r_1.\ \nu r_2.\{^{\mathsf{pk}(skE)}/_{xk}\} \mid \{^{a}/_{x'}, {}^{b}/_{y'}, {}^{b}/_{z'}\} \mid$$
$$\{^{\langle id_1, \mathsf{aenc}(\mathsf{pk}(skE), r_1, b)\rangle}/_{x}, {}^{\langle id_2, \mathsf{aenc}(\mathsf{pk}(skE), r_2, a)\rangle}/_{y}, {}^{\langle id_3, \mathsf{aenc}(\mathsf{pk}(skE), r_1, b)\rangle}/_{z}, \}$$

The two frames ϕ and ψ are not statically equivalent for the equational theory $\mathsf{E}_{\mathsf{enc}}$. Indeed, consider for example $M = z'$ and $N = a$, we have $M\phi = a = N\phi$ but $M\psi = b \neq N\psi$. Therefore, $\phi \not\sim_{\mathsf{E}_{\mathsf{enc}}} \psi$.

The active counterpart of static equivalence is trace equivalence. Intuitively, two processes A and B are indistinguishable to an attacker if any execution of A can be matched to an execution of B that is equal for their observable actions and such that the corresponding sequences of sent messages are statically equivalent.

Definition 2 (Trace equivalence). *Given an equational theory* E *two processes* A *and* B *are trace equivalent, denoted* $A \overset{\mathsf{tr}}{\approx}_{\mathsf{E}} B$*, if for any trace* $(tr_A, \phi_A) \in \mathsf{trace}_{\mathsf{E}}(A)$ *there is a corresponding trace* $(tr_B, \phi_B) \in \mathsf{trace}_{\mathsf{E}}(B)$ *such that* $tr_A = tr_B$ *and* $\phi_A \sim_{\mathsf{E}} \phi_B$ *(and reciprocally).*

Example 5. We consider the Helios system H' with two honest voters and one dishonest voter where one honest voter casts the vote b while the other one casts the vote a.

$$H' \overset{\mathsf{def}}{=} \nu skE.\ \nu auth_1.\ \nu auth_2.$$
$$\overline{bb}\langle \mathsf{pk}(skE)\rangle.\ (V(auth_1, id_1, b) \mid V(auth_2, id_2, a) \mid BB(auth_1, auth_2, auth_3))$$

Let (tr, ϕ) be the trace corresponding to the execution of H described in Example 3 where $\phi = \varphi(H_2) = H_2$ (as defined in Example 3) and $tr = \nu xk.\ \overline{bb}\langle xk\rangle \cdot \nu x.\ \overline{bb}\langle x\rangle \cdot \nu y.\ \overline{bb}\langle y\rangle \cdot auth_3(\langle id_3, x\rangle) \cdot \nu z.\ \overline{bb}\langle z\rangle \cdot \nu x'.\ \overline{bb}\langle x'\rangle \cdot \nu y'.\ \overline{bb}\langle y'\rangle \cdot \nu z'.\ \overline{bb}\langle z'\rangle$. Then $(tr, \phi) \in \mathsf{trace}_{\mathsf{E}_{\mathsf{enc}}}(H)$ and for any $(tr, \phi') \in \mathsf{trace}_{\mathsf{E}_{\mathsf{enc}}}(H')$, it is easy to check that we have $\phi \not\sim_{\mathsf{E}_{\mathsf{enc}}} \phi'$. (In fact, $\phi' = \psi$ from Example 4.) Therefore, $H \overset{\mathsf{tr}}{\not\approx}_{\mathsf{E}_{\mathsf{enc}}} H'$

Intuitively, if the dishonest voter's strategy is to replay the first voter's vote, then he would cast a vote of the form $\langle id_3, \mathsf{aenc}(\mathsf{pk}(skE), r_1, a)\rangle$ in the system H while he would cast a vote of the form $\langle id_3, \mathsf{aenc}(\mathsf{pk}(skE), r_1, b)\rangle$ in the system H'. Once the result is published, an attacker would be then able to distinguish H from H' since the tally in H is $\{a, b, a\}$ while it is $\{b, a, b\}$ in H'. This corresponds exactly to the replay attack against Helios, explained in [9].

3 Forward and everlasting indistinguishability

In this section we introduce and illustrate our definitions of forward and everlasting indistinguishability. In the next section we will show how the here presented definitions can be used to define practical everlasting privacy in electronic voting.

3.1 Definitions of forward and everlasting indistinguishability

From now on we suppose that Σ is a signature and that E_0 and E_1 are equational theories over Σ. We want to model that an attacker may interact with a protocol today and store some data which may be exploited in the future when his computational power has increased. We model the fact that the attacker's computational power may change by using two different equational theories: E_0 models the attacker's capabilities while interacting with the protocol at the time of the election, while E_1 models his capabilities when exploiting the published data in the future when the strength of cryptography has been eroded.

We also argue that in many situations it is reasonable to suppose that the attacker does not store all of the data that was sent over the network. We will therefore consider some channels to be *everlasting*: data sent over such channels will remain in the attacker's knowledge for future analysis while other data will be "forgotten" and can only be used during the interaction with the protocol. Typically, everlasting channels are media such as web-pages published on the Internet (that can easily be accessed by anyone, for a rather long period of time) while public but non-everlasting channels can be communications over the Internet, which can be recorded only by the active and costly involvement of an attacker.

In order to reason about data that has been sent on certain channels we introduce the following notation. Let P be a process, $\mathcal{C}$ a set of channels (corresponding to the everlasting channels), and $tr = (\alpha_1 \cdot \alpha_2 \cdot \ldots \cdot \alpha_n, \varphi) \in \mathsf{trace}_E(P)$ a trace of P. We define the set of variables in the domain of φ corresponding to terms sent on channels in $\mathcal{C}$ as $\mathcal{V}_{\mathcal{C}}(\alpha_1 \cdot \alpha_2 \cdot \ldots \cdot \alpha_n) = \{x \mid c \in \mathcal{C},\ 1 \leq i \leq n,\ \alpha_i = \nu x.\, \overline{c}\langle x \rangle\}$ and denote by $\phi_{\mathcal{V}}(P_n)$ the substitution $\phi(P_n)$ whose domain is restricted to the set of variables $\mathcal{V}$.

Two processes A and B are said to be forward indistinguishable if, informally, an attacker cannot observe the difference between A and B being given the computational power modeled by E_1 (where it can break keys for example), but for executions that happened in the past, that is over E_0 (the standard theory) and observing only the information that was passed through everlasting channels.

Definition 3 (Forward indistinguishability). *Let A and B be two closed extended processes and $\mathcal{C}$ a set of channels. We define $A \overset{\mathsf{fwd}}{\sqsubseteq}{}^{\mathcal{C}}_{\mathsf{E}_0,\mathsf{E}_1} B$, if for every trace $(\alpha_1 \cdot \alpha_2 \cdots \alpha_n, \varphi_A) \in \mathsf{trace}_{\mathsf{E}_0}(A)$ there exists φ_B s.t. $(\alpha_1 \cdot \alpha_2 \cdots \alpha_n, \varphi_B) \in \mathsf{trace}_{\mathsf{E}_0}(B)$*

$$\textit{and} \quad \phi_{A\mathcal{V}} \sim_{\mathsf{E}_1} \phi_{B\mathcal{V}}.$$

where $\mathcal{V} = \mathcal{V}_{\mathcal{C}}(\alpha_1 \cdot \alpha_2 \cdots \alpha_n)$. A and B are forward indistinguishable w.r.t. $\mathcal{C}$, E_0 *and* E_1*, denoted $A \overset{\mathsf{fwd}}{\approx}{}^{\mathcal{C}}_{\mathsf{E}_0,\mathsf{E}_1} B$, if $A \overset{\mathsf{fwd}}{\sqsubseteq}{}^{\mathcal{C}}_{\mathsf{E}_0,\mathsf{E}_1} B$ and $B \overset{\mathsf{fwd}}{\sqsubseteq}{}^{\mathcal{C}}_{\mathsf{E}_0,\mathsf{E}_1} A$.*

Note that in the above definition we only check equivalence of messages that were sent on channels in the set $\mathcal{C}$. We may also require that A and B are indistinguishable in the standard way (over E_0). Standard indistinguishability and forward indistinguishability yield *everlasting indistinguishability.*

Definition 4 (Everlasting indistinguishability). *Let A and B be two closed extended processes, $\mathcal{C}$ a set of channels. A and B are* everlasting indistinguishable w.r.t. $\mathcal{C}$, E_0 *and* E_1*, denoted $A \overset{\mathsf{ev}}{\approx}{}^{\mathcal{C}}_{\mathsf{E}_0,\mathsf{E}_1}$ if*

1. $A \overset{\mathsf{tr}}{\approx}_{\mathsf{E}_0} B$, i.e. *$A$ and B are trace equivalent* w.r.t. E_0*; and*
2. $A \overset{\mathsf{fwd}}{\approx}{}^{\mathcal{C}}_{\mathsf{E}_0,\mathsf{E}_1} B$, i.e. *$A$ and B are forward indistinguishable* w.r.t. $\mathcal{C}$, E_0 *and* E_1.

3.2 Examples

We illustrate the above definitions on a simple RFID protocol. In the context of RFID systems, forward secrecy is often a desired property: even if an RFID tag has been tampered with, and its secrets have been retrieved by a malicious entity, its past transactions should remain private. This can be seen as a form of everlasting security requirement. Indeed, RFID tags being devices vulnerable to tampering, one would like to make sure that when an intruder gains access to an honest device, he is not able to trace back the movements of the tag. Such tampering can be modelled by the following equational theory $\mathsf{E}_{\mathsf{break}}$, that gives direct access to keys.

$$\mathsf{E}_{\mathsf{break}} = \left\{ \begin{array}{r} \mathsf{break}_{\mathsf{aenc}}(\mathsf{aenc}(\mathsf{pk}(x), y, z)) = x \\ \mathsf{break}_{\mathsf{senc}}(\mathsf{senc}(x, y, z)) = x \end{array} \right\}$$

We also use this equational theory later to model that in 20 or 30 years an adversary will be able to break nowadays encryption keys.

Consider the following toy RFID protocol

$$\begin{aligned} session &= \nu r.\ \bar{c}\langle \mathsf{enc}(k, r, id)\rangle \\ tag &= \nu k.\ \nu id.\ !session \\ system &= !tag \end{aligned}$$

where a tag identifies itself to a reader by sending its tag identifier id encrypted with a long-term symmetric key shared between the tag and the reader.

We can model unlinkability as being the property that an attacker cannot distinguish the situation where the same tag is used in several sessions from the situation where different tags are used. Formally unlinkability is modelled as the trace equivalence:

$$system \overset{\mathsf{tr}}{\approx}_{\mathsf{E_{enc}}} system'$$

where

$$system' = !\nu k.\nu id.\ session.$$

Intuitively, this protocols satisfies unlinkability only as long as the keys are not leaked. Indeed, since each identification uses a different random in the encrypted message sent to the reader, each of the sent messages is different and looks like a random message to the intruder. However, $system$ and $system'$ are not forward indistinguishable when considering a theory E_1 which allows to break keys, i.e.,

$$system \overset{\mathsf{fwd}}{\not\approx}{}^{\{c\}}_{\mathsf{E_{enc}},\mathsf{E_{enc}}\cup\mathsf{E_{break}}} system'$$

where $\mathsf{E_{enc}}$ is the equational theory introduced in Example 1. Indeed, once the key k of a tag is obtained by the intruder, he can retrieve the identity behind each blob he has seen on channel c, and thus distinguish the set of messages obtained by an execution of $system$ where the same tag executes at least two sessions, from the set of messages obtained by the corresponding execution of $system'$ where each tag has executed at most one session. Thus this protocol does not satisfy the stronger requirement of everlasting indistinguishability either:

$$system \overset{\mathsf{ev}}{\not\approx}{}^{\{c\}}_{\mathsf{E_{enc}},\mathsf{E_{enc}}\cup\mathsf{E_{break}}} system'$$

4 Application to practical everlasting privacy

We model a practical version of everlasting privacy in voting protocols based on everlasting indistinguishability.

4.1 Definition of practical everlasting privacy

We first recall the definition of vote privacy introduced in [15].

Definition 5 (Vote privacy). *Let* $\mathsf{VP}(\mathsf{v}_1, \mathsf{v}_2)$ *be an extended process with two free variables* v_1, v_2. $\mathsf{VP}(\mathsf{v}_1, \mathsf{v}_2)$ *respects vote privacy for an equational theory* E *if*

$$\mathsf{VP}(a, b) \overset{\mathsf{tr}}{\approx}_{\mathsf{E}} \mathsf{VP}(b, a)$$

Intuitively, the free variables refer to the votes of two honest voters id_1 and id_2. Then this equivalence ensures that an attacker cannot distinguish the situations where voter id_1 voted for candidate a and voter id_2 voted for candidate b, from the situation where the voters swapped their votes, i.e., voter id_1 voted for candidate b and voter id_2 voted for candidate a.

Example 6. Let $\mathsf{Helios}(v_1, v_2)$ be the process

$$\begin{aligned} &\nu skE.\ \nu auth_1.\ \nu auth_2. \\ &\overline{bb}\langle \mathsf{pk}(skE)\rangle.\ (V(auth_1, id_1, v_1) \mid V(auth_2, id_2, v_2) \mid BB(auth_1, auth_2, auth_3)) \end{aligned}$$

where processes V and BB are defined in Example 2.

In Example 5, when we illustrated trace equivalence we showed that Helios does not satisfy vote privacy due to a vote replay attack discovered in [9].

A simple fix of the attack consists in weeding duplicates. The corresponding tally is

$$\begin{aligned} T' \overset{\mathsf{def}}{=}\ & c(x').c(y').c(z'). \\ & \quad \mathsf{if}\ \mathsf{snd}(x') \neq \mathsf{snd}(y')\ \wedge\ \mathsf{snd}(x') \neq \mathsf{snd}(z') \wedge \mathsf{snd}(y') \neq \mathsf{snd}(z')\ \mathrm{then} \\ & \qquad \overline{bb}\langle \mathsf{adec}(skE, \mathsf{snd}(x'))\rangle \mid \overline{bb}\langle \mathsf{adec}(skE, \mathsf{snd}(y'))\rangle \mid \overline{bb}\langle \mathsf{adec}(skE, \mathsf{snd}(z'))\rangle \end{aligned}$$

In other words, the tally is performed only if there are no duplicates amongst the cast votes. We define the voting protocol $\mathsf{Helios}^{\mathsf{noreplay}}$ as Helios but with the revised version T' of the tally. Using the tools ProVerif and AKISS we have shown that this protocol satisfies vote privacy.

$$\mathsf{Helios}^{\mathsf{noreplay}}(a, b) \overset{\mathsf{tr}}{\approx}_{\mathsf{E}_{\mathsf{enc}}} \mathsf{Helios}^{\mathsf{noreplay}}(b, a)$$

The above definition of vote privacy does however not take into account that most cryptographic schemes rely on computational assumptions and may be broken in the future. In order to protect the secrecy of votes against such attacks in the future we propose a stronger definition based on forward indistinguishability.

Definition 6 (Everlasting vote privacy). *Let* $\mathsf{VP}(\mathsf{v}_1, \mathsf{v}_2)$ *be an extended process with two free variables* v_1, v_2. $\mathsf{VP}(v_1, v_2)$ *satisfies* everlasting privacy *w.r.t. a set of channels* $\mathcal{C}$ *and equational theories* E_0 *and* E_1*, if*

$$\mathsf{VP}(a, b) \overset{\mathsf{ev}}{\approx}{}^{\mathcal{C}}_{\mathsf{E}_0,\mathsf{E}_1} \mathsf{VP}(b, a)$$

We note that everlasting vote privacy is strictly stronger than vote privacy as it requires trace equivalence of $\mathsf{VP}(a, b)$ and $\mathsf{VP}(b, a)$ (which is exactly vote privacy) and additionally forward indistinguishability of these processes. Our definition is parametric with respect to the equational theories and the channels we suppose to be everlasting. The equational theory E_1 allows us to exactly specify what a future attacker may be able to break. The set of everlasting channels $\mathcal{C}$ allows us to specify what data a future attacker has access to. When $\mathcal{C}$ corresponds to all channels we typically get a requirement which is too strong for practical purposes. We argue that it is reasonable to suppose that in, say 50 years, an attacker does not have access to the transmissions between individual voters and the system while a bulletin board published on the Internet could easily have been stored.

4.2 Examples

Helios with identities As discussed In Example 6, $\mathsf{Helios}^{\mathsf{noreplay}}$ does satisfy vote privacy. However, this protocol does not satisfy everlasting vote privacy with $\mathsf{E}_0 = \mathsf{E}_{\mathsf{enc}}$, $\mathsf{E}_1 = \mathsf{E}_{\mathsf{enc}} \cup \mathsf{E}_{\mathsf{break}}$ and $\mathcal{C} = \{bb\}$. Intuitively, this is due to the fact that a future attacker can break encryption and link the recovered vote to the identity submitted together with the cast ballot. Formally, we can show that

$$\mathsf{Helios}^{\mathsf{noreplay}}(a,b) \overset{\mathsf{fwd}}{\not\approx} \mathsf{Helios}^{\mathsf{noreplay}}(b,a)$$

Consider the trace $(\nu xk.\ \overline{bb}\langle xk\rangle \cdot \nu x.\ \overline{bb}\langle x\rangle \cdot \nu y.\ \overline{bb}\langle y\rangle, \varphi_A) \in \mathsf{trace}_{\mathsf{E}_{\mathsf{enc}}}(\mathsf{Helios}^{\mathsf{noreplay}}(a,b))$ where

$$\begin{aligned}\varphi_A = \nu skE, r_1, r_2.\{ &\ {}^{\mathsf{pk}(skE)}/_{xk},\\ &\ {}^{\langle id_1, \mathsf{aenc}(\mathsf{pk}(skE), r_1, a)\rangle}/_{x},\\ &\ {}^{\langle id_2, \mathsf{aenc}(\mathsf{pk}(skE), r_2, b)\rangle}/_{y}\ \}\end{aligned}$$

Traces $(\nu xk.\ \overline{bb}\langle xk\rangle \cdot \nu x.\ \overline{bb}\langle x\rangle \cdot \nu y.\ \overline{bb}\langle y\rangle, \varphi_B) \in \mathsf{trace}_{\mathsf{E}_{\mathsf{enc}}}(\mathsf{Helios}^{\mathsf{noreplay}}(b,a))$ are either such that

$$\begin{aligned}\varphi_B \equiv \nu skE, r_1, r_2.\{ &\ {}^{\mathsf{pk}(skE)}/_{xk},\\ &\ {}^{\langle id_1, \mathsf{aenc}(\mathsf{pk}(skE), r_1, b)\rangle}/_{x},\\ &\ {}^{\langle id_2, \mathsf{aenc}(\mathsf{pk}(skE), r_2, a)\rangle}/_{y}\ \}\end{aligned}$$

or

$$\begin{aligned}\varphi_B \equiv \nu skE, r_1, r_2.\{ &\ {}^{\mathsf{pk}(skE)}/_{xk},\\ &\ {}^{\langle id_2, \mathsf{aenc}(\mathsf{pk}(skE), r_1, a)\rangle}/_{x},\\ &\ {}^{\langle id_1, \mathsf{aenc}(\mathsf{pk}(skE), r_2, b)\rangle}/_{y}\ \}\end{aligned}$$

In both cases we have that $\varphi_A \not\sim_{\mathsf{E}_{\mathsf{enc}} \cup \mathsf{E}_{\mathsf{break}}} \varphi_B$. In the first case this is witnessed by the test $M = a$ and $N = \mathsf{break}_{\mathsf{aenc}}(\mathsf{snd}(x))$ as

$$M\varphi_A = a =_{\mathsf{E}_{\mathsf{enc}} \cup \mathsf{E}_{\mathsf{break}}} N\varphi_A \quad \text{but} \quad M\varphi_B = a \neq_{\mathsf{E}_{\mathsf{enc}} \cup \mathsf{E}_{\mathsf{break}}} b =_{\mathsf{E}_{\mathsf{enc}} \cup \mathsf{E}_{\mathsf{break}}} N\varphi_B$$

In the second case non equivalence is witnessed by the test $M = id_1$ and $N = \mathsf{fst}(x)$.

Helios without identities As we just saw $\mathsf{Helios}^{\mathsf{noreplay}}$ does not satisfy everlasting privacy. This is due to the fact that encrypted votes are published together with the identity of the voter on the bulletin board. A simple variant (used e.g. in Louvain for student elections) consists in publishing the encrypted vote without the identity of the voter. We define $\mathsf{Helios}^{\mathsf{noid}}$ as $\mathsf{Helios}^{\mathsf{noreplay}}$ but redefining the process BB' as

$$\begin{aligned}BB'(a_1, a_2, a_3) \overset{\mathsf{def}}{=} \nu c.\ a_1(x).\ \overline{bb}\langle \mathsf{snd}(x)\rangle.\ \overline{c}\langle x\rangle \mid a_2(y).\ \overline{bb}\langle \mathsf{snd}(y)\rangle.\ \overline{c}\langle y\rangle \mid \\ a_3(z).\ \overline{bb}\langle \mathsf{snd}(z)\rangle.\ \overline{c}\langle z\rangle \mid T'\end{aligned}$$

where T' is as defined at Example 6. As we shall see in Section 6, we prove everlasting privacy of $\mathsf{Helios}^{\mathsf{noid}}$ w.r.t $\mathsf{E}_{\mathsf{enc}}$, $\mathsf{E}_{\mathsf{break}}$ and everlasting channel bb, using (adaptations of) ProVerif and AKISS.

5 Modeling commitments

Commitment schemes allow a sender to commit to a value v while keeping this value hidden until an 'opening' phase, where the sender reveals v to the receiver of the commitment $\mathsf{commit}(v)$. The receiver should then be able to verify that the revealed value is indeed the one used to compute $\mathsf{commit}(v)$, and in that sense that the sender had indeed committed to the revealed value. The two main security properties of such schemes are *binding* (the sender can't claim that $\mathsf{commit}(v)$ is a commitment to some $v' \neq v$), and *hiding* (the receiver can't deduce v from $\mathsf{commit}(v)$). These two properties can hold 'perfectly' or merely 'computationally'. It is known that there are no commitment schemes which are both perfectly hiding and perfectly binding, so one has to choose between perfectly hiding and computationally binding (PHCB) and perfectly binding and computationally hiding (PBCH). In this section, we characterize in our formal model what it means for a primitive to be PHCB and PBCH. We also give equational theories to model such primitives, which we then use for the verification of two voting protocols that rely on such primitives to ensure everlasting vote privacy.

5.1 Modeling hiding and binding cryptographic primitives

PBCH primitives Informally, an n-ary function f is *perfectly binding* if the inputs are totally determined by the output. In other words, f is perfectly binding if it admits no collisions. It is *computationally hiding* if it is hard to retrieve the inputs from the output.

To model a PBCH primitive f using the applied pi calculus, we introduce two equational theories $\mathsf{E}_0^{\mathsf{f}}$ and $\mathsf{E}_1^{\mathsf{f}}$ over the signature $\Sigma = \{\mathsf{f}, \mathsf{break}_{\mathsf{f}}^1, \ldots, \mathsf{break}_{\mathsf{f}}^n\}$, such that no equation of the form

$$\mathsf{f}(M_1, \ldots, M_n) =_{\mathsf{E}} \mathsf{f}(N_1, \ldots, N_n)$$

is derivable, where $(M_1, \ldots, M_n) \neq_{\mathsf{E}} (N_1, \ldots, N_n)$ and $\mathsf{E} \in \{\mathsf{E}_0^{\mathsf{f}}, \mathsf{E}_1^{\mathsf{f}}\}$; and that the equation

$$\mathsf{break}_{\mathsf{f}}^i(\mathsf{f}(v_1, \ldots, v_n)) =_{\mathsf{E}_1^{\mathsf{f}}} v_i.$$

is derivable. As before, $\mathsf{E}_0^{\mathsf{f}}$ models the capabilities of a computationally bounded attacker interacting with the protocol, while $\mathsf{E}_1^{\mathsf{f}}$ models the capabilities of a computationally unbounded attacker in the future.

Example 7. A trivial example of a perfectly binding function is the identity function id. However, id is not hiding.

Example 8. An example of a PBCH primitive is the ElGamal public key derivation function. Given multiplicative cyclic group G of order q with generator g, to generate a private and public key pair Alice does the following:

1. she chooses at random her private key $sk \in \{1, \ldots, q-1\}$,
2. she computes and publishes her public key $\mathsf{pk}_{G,g,q}(sk) = g^{sk}$.

The secret key sk is totally determined by the public key $\mathsf{pk}_{G,g,q}(sk) = g^{sk}$. It is however as hard to find sk from $\mathsf{pk}_{G,g,q}(sk)$ as it is to compute discrete logarithms.

Thus, to reason about protocols relying on ElGamal encryption we consider the following equational theories over the signature $\{\mathsf{aenc}_{G,g,q}, \mathsf{adec}_{G,g,q}, \mathsf{pk}_{G,g,q}, \mathsf{break}_{\mathsf{pk}_{G,g,q}}\}$ (we omit the subscripts G, g, q for readability):

$$\mathsf{E}_0^{\mathsf{ElGamal}} = \{\mathsf{adec}(xk, \mathsf{aenc}(\mathsf{pk}(xk), xr, xm)) = xm\}$$

$$\mathsf{E}_1^{\mathsf{ElGamal}} = \left\{ \begin{array}{l} \mathsf{adec}(xk, \mathsf{aenc}(\mathsf{pk}(xk), xr, xm)) = xm \\ \mathsf{break}_{\mathsf{pk}}(\mathsf{pk}(xk)) = xk \end{array} \right\}$$

The function $\mathsf{pk}_{G,g,q}$ is PBCH. Note however that the encryption algorithm $\mathsf{aenc}_{G,g,q}$ is not PBCH, since it is not perfectly binding. Indeed, given the parameters G, q, and g, to encrypt the message m with the public key g^{sk}, Alice would

1. pick a random $r \in \{0, \ldots, q-1\}$ and comput $c_1 = g^r$;
2. compute the secret shared key $s = (g^{sk})^r$; and
3. computer $c_2 = m.s$

The computed ciphertext is then $\mathsf{aenc}(\mathsf{pk}(sk), r, m) = (c_1, c_2) = (g^r, m.(g^{sk})^r)$. Hence, for any public key $\mathsf{pk}(sk') = g^{sk'}$, there exists a message $m' = m.(g^{sk})^r/(g^{sk'})^r$ such that $\mathsf{aenc}(\mathsf{pk}(sk), r, m) = \mathsf{aenc}(\mathsf{pk}(sk'), r, m')$. Thus, ElGamal encryption is not perfectly binding.

PHCB primitives Informally, an n-ary function f is perfectly hiding if given the output, it is impossible to retrieve any of the inputs. So even enumerating all the possible inputs shouldn't allow one to retrieve the inputs from the output of the function. But this implies that f should admit collisions for each possible input. On the other hand, f is computationally binding if it is computationally not feasible to find such collisions.

Example 9. Any constant function $f(x_1, \ldots, x_n) = c$ is obviously perfectly hiding but not computationally binding. The $\oplus$ function is also perfectly hiding since for all $z = x \oplus y$

– for all x', we have that $y' = z \oplus x'$ is such that $x \oplus y = x' \oplus y'$; and
– for all y'', we have that $x'' = z \oplus y''$ is such that $x \oplus y = x'' \oplus y''$.

However, it is not computationally binding since it is easy to compute x'' and y'.

Example 10. Pedersen commitments are PHCB. The Pedersen commitment over a cyclic group G of order q and two generators $h, g \in G$ such that $\log_g h$ is not known is the function $P_{h,g}^G(x, y) = h^x \cdot g^y (\bmod q)$. Pedersen commitments are perfectly hiding since for all $z = P_{h,g}^G(x, y)$,

– for all x', we have that $y' = y + (x - x') \cdot \log_g h \bmod q$ is such that $P_{h,g}^G(x, y) = P_{h,g}^G(x', y')$;
– for all y'', we have that $x'' = x + (y - y'') \cdot \log_h g \bmod q$ is such that $P_{h,g}^G(x, y) = P_{h,g}^G(x'', y'')$.

but they are computationally binding because finding these x'' and y' is as hard as computing discrete logarithms.

To reason about protocols relying on Pedersen commitments using the applied pi calculus, we introduce the function symbols $\mathsf{forge}^1_{\mathsf{Ped}}$, and $\mathsf{forge}^2_{\mathsf{Ped}}$ and the two following equational theories

$$\mathsf{E}_0^{\mathsf{Ped}} = \emptyset$$

$$\mathsf{E}_1^{\mathsf{Ped}} = \left\{ \begin{array}{l} \mathsf{Ped}(\mathsf{forge}^1_{\mathsf{Ped}}(v,y'),y') = v \\ \mathsf{Ped}(x',\mathsf{forge}^2_{\mathsf{Ped}}(v,x')) = v \\ \mathsf{forge}^1_{\mathsf{Ped}}(\mathsf{Ped}(x,y),y) = x \\ \mathsf{forge}^2_{\mathsf{Ped}}(\mathsf{Ped}(x,y),x) = y \\ \mathsf{forge}^1_{\mathsf{Ped}}(v,\mathsf{forge}^2_{\mathsf{Ped}}(v,x)) = x \\ \mathsf{forge}^2_{\mathsf{Ped}}(v,\mathsf{forge}^1_{\mathsf{Ped}}(v,y)) = y \end{array} \right\}$$

For the first equation, suppose $v = \mathsf{Ped}(x,y)$, and we have some y'; then $\mathsf{forge}^1_{\mathsf{Ped}}$ allows us to compute a value $x' = \mathsf{forge}^1_{\mathsf{Ped}}(v,y')$ such that $v = \mathsf{Ped}(x',y')$. The second equation is similar. The third and fourth equation allow us to recover one of the arguments, given that the other argument is known. In other words the third equation expresses that when forging $x' = \mathsf{forge}^1_{\mathsf{Ped}}(v,y)$ and $v = \mathsf{Ped}(x,y)$ then we must have that $x' = x$, and similarly for the fourth equation. The fifth and sixth equations are also seen to be mathematically valid, given that $\mathsf{forge}^1_{\mathsf{Ped}}(v,y)$ and $\mathsf{forge}^2_{\mathsf{Ped}}(v,x)$ respectively model the terms $\log_g(v/h^y)$ and $\log_h(v/g^x)$.

5.2 Applications: electronic voting protocols and everlasting privacy

Pedersen commitments have been used in several voting protocols for achieving everlasting privacy. In particular we study the protocol by Moran and Naor [18] and a recent version of Helios [12] based on Pedersen commitments.

Moran-Naor protocol Moran and Naor [18] designed a protocol to be used with voting machines in a polling station. The protocol aims to achieve both verifiability and everlasting privacy. From a high level point of view the protocol works as follows.

1. The voter enters his vote into the voting machine inside the voting booth. The machine then computes a Pedersen commitment to this vote and provides a zero knowledge proof to the voter that the computed value commits to the voter's choice. The commitment is then published on a bulletin board so that the voter can verify the presence of his ballot.
2. After all ballots have been cast, the votes are published (in random order) on the bulletin board together with a zero knowledge proof asserting that the published votes correspond to the votes of the published commitments.

As we are only interested in privacy and not verifiability we ignore the zero knowledge proofs in our modeling and simply represent the protocol by the process

$$\begin{aligned} \mathsf{MoranNaor}(v_1,v_2) &\overset{\mathsf{def}}{=} \nu priv_1.\ \nu priv_2. \\ &V(priv_1,v_1) \mid V(priv_2,v_2) \mid \nu c.(DRE(priv_1,priv_2,priv_3) \mid T) \end{aligned}$$

where

$$\begin{aligned}V(priv, v) &\overset{\mathsf{def}}{=} \overline{priv}\langle v\rangle\\ DRE(p_1, p_2, p_3) &\overset{\mathsf{def}}{=} p_1(x_1).\nu r_1.\overline{bb}\langle \mathsf{Ped}(x_1, r_1)\rangle.\bar{c}\langle x_1\rangle \mid\\ &\quad\ p_2(x_2).\nu r_2.\overline{bb}\langle \mathsf{Ped}(x_2, r_2)\rangle.\bar{c}\langle x_2\rangle \mid\\ &\quad\ p_3(x_3).\nu r_3.\overline{bb}\langle \mathsf{Ped}(x_3, r_3)\rangle.\bar{c}\langle x_3\rangle\\ T &= c(y_1).\overline{bb}\langle y_1\rangle \mid c(y_2).\overline{bb}\langle y_2\rangle \mid c(y_3).\overline{bb}\langle y_3\rangle\end{aligned}$$

As the voter enters his vote in a private ballot booth, we have modelled this communication on a private channel. We have been able to show that MoranNaor verifies everlasting privacy with respect to the channel *bb* and the equational theories introduced for Pedersen commitments.

Helios with Pedersen commitments In [12], the authors propose a version of the Helios voting system that provides everlasting vote privacy *w.r.t.* the bulletin board. They rely for this on Pedersen commitments. In this section, we present this new version of the Helios system.

1. The voter V chooses her candidate v and commits to it by generating a random number r and computing the Pedersen commitment $\mathsf{Ped}(r, v)$. She then separately encrypts the decommitment values r and v using the public key $\mathsf{pk}(skE)$ of the election; and casts her commitment together with the encrypted decommitment values and her identity on a private authenticated channel. Upon reception of the ballot, the Bulletin Board (BB) publishes on a public web page the commitment $\mathsf{Ped}(r, v)$ (after having checked that V is entitled to vote).
2. Once the voting phase is over, the ballots (*i.e.* the commitments together with the encrypted decommitment values) are shuffled and rerandomized through mixnets. The random permutation of the rerandomized ballots is published on the public webpage (together with a zero knowledge proof of correct reencryption and mixing).
3. Finally, the authorities decrypt the rerandomized and shuffled decommitment values and the BB publishes them.

The voter can be modelled by the following process:

$$\begin{aligned}V(id, auth, v) &\overset{\mathsf{def}}{=} \nu s.\nu rv.\nu rs.\\ &\overline{auth}\langle\langle id, \langle \mathsf{Ped}(s, v), \langle \mathsf{aenc}(\mathsf{pk}(skE), rv, v), \mathsf{aenc}(\mathsf{pk}(skE), rs, s)\rangle\rangle\rangle\rangle\end{aligned}$$

She sends to the BB on the private authenticated channel $authCh$, her commitment $\mathsf{Ped}(s, v)$ to vote v, together with her identity and the encrypted decommitment values $\mathsf{aenc}(\mathsf{pk}(skE), rv, v)$, $\mathsf{aenc}(\mathsf{pk}(skE), rs, s)$.

The ballot box publishes her commitment for verifiability purposes. After having received all votes, the BB publishes the votes in a random order through the process T.

$$\begin{aligned}BB(a_1, a_2, a_3) &\overset{\mathsf{def}}{=} a_1(x).\ \overline{bb}\langle\langle \mathsf{fst}(x), \mathsf{fst}(\mathsf{snd}(x))\rangle\rangle.\ \bar{c}\langle x\rangle \mid\\ &\quad\ a_2(y).\ \overline{bb}\langle\langle \mathsf{fst}(x), \mathsf{fst}(\mathsf{snd}(x))\rangle\rangle.\ \bar{c}\langle x\rangle \mid\\ &\quad\ a_3(z).\ \bar{c}\langle z\rangle \mid T\end{aligned}$$

$$
\begin{aligned}
T \overset{\mathsf{def}}{=} c(x).\, c(y).\, c(z).\text{if } & \mathsf{fst}(\mathsf{snd}(\mathsf{snd}(x))) \neq \mathsf{fst}(\mathsf{snd}(\mathsf{snd}(z))) \\
& \wedge \mathsf{fst}(\mathsf{snd}(\mathsf{snd}(y))) \neq \mathsf{fst}(\mathsf{snd}(\mathsf{snd}(z))) \\
& \wedge \mathsf{fst}(\mathsf{snd}(\mathsf{snd}(x))) \neq \mathsf{fst}(\mathsf{snd}(\mathsf{snd}(y))) \text{ then} \\
& \overline{bb}\langle \mathsf{adec}(skE, \mathsf{fst}(\mathsf{snd}(\mathsf{snd}(x))))\rangle \mid \\
& \overline{bb}\langle \mathsf{adec}(skE, \mathsf{fst}(\mathsf{snd}(\mathsf{snd}(y))))\rangle \mid \\
& \overline{bb}\langle \mathsf{adec}(skE, \mathsf{fst}(\mathsf{snd}(\mathsf{snd}(z))))\rangle
\end{aligned}
$$

Finally we can define the voting protocol $\mathsf{Helios}^{\mathsf{Ped}}$ as

$$
\begin{aligned}
&\mathsf{Helios}^{\mathsf{Ped}}(v1, v2) \overset{\mathsf{def}}{=} \nu skE.\, \nu auth_1.\, \nu auth_2. \\
&\quad \overline{bb}\langle \mathsf{pk}(skE)\rangle.\, (V(auth_1, id_1, v_1) \mid V(auth_2, id_2, v_2) \mid BB(auth_1, auth_2, auth_3))
\end{aligned}
$$

which verifies everlasting privacy with respect to the channel bb and the previously introduced equational theories.

6 Tool support for everlasting indistinguishability

In order to verify everlasting indistinguishability on the examples presented in the previous section we have adapted two tools for automated verification of equivalence properties, AKiSs [6] and ProVerif [5]. The two tools have shown themselves to be complementary and the results obtained using the tools are summarized in Figure 3.

AKiSs. AKiSs is a recent tool that has been designed to automatically prove trace equivalence by translating processes into Horn clauses and using a dedicated resolution algorithm. More precisely it can both under- and over-approximate trace equivalence in the case of a bounded number of sessions, i.e. for processes without replication. The tool has currently two limitations: it does not support private channels, or else branches in conditionals. However, it is able to deal with a wide range of equational theories, including the theory for Pedersen commitments introduced in the previous section.

We have adapted the tool in order to check forward indistinguishability and adapted the syntax to declare everlasting channels and an everlasting equational theory. More precisely we implemented an algorithm to check an under-approximation of forward indistinguishability, yielding a proof of forward indistinguishability whenever the tool responds positively. While false attacks are possible, we did not encounter any in our case studies.

Absence of support for private channels and else branches required us to adapt some of the examples. In particular we rewrote the processes by directly *inlining* private communications, which in the examples maintained the same set of traces, hence preserving everlasting indistinguishability. The weeding operation in $\mathsf{Helios}^{\mathsf{noreplay}}$, $\mathsf{Helios}^{\mathsf{noid}}$ and $\mathsf{Helios}^{\mathsf{ped}}$ requires the use of an else branch. We encoded a different weeding procedure using cryptographic proofs of knowledge. While the vote replay attack on the simple Helios protocol is found in less than 10 seconds, the verification of other examples ranged from a few minutes to several hours. While attempting to verify $\mathsf{Helios}^{\mathsf{ped}}$ the tool ran out of memory and we were only able to verify a version of $\mathsf{Helios}^{\mathsf{ped}}$ with two honest voters and no dishonest voter. As the tool is still in a prototype status we are confident that future optimizations will allow the tool to scale up to this kind of protocols. The tool and example files are available at `https://github.com/ciobaca/akiss`.

ProVerif. The ProVerif tool [4] is an automatic cryptographic protocol verifier. It is based on the representation of protocols by Horn clauses and relies on several approximations. ProVerif can handle several types of properties and in particular equivalence based properties [5] like the privacy-type ones which we are interested in this work. Moreover, ProVerif can handle many different cryptographic primitives, including Pedersen commitments as our case studies show.

The ProVerif tool works by translating biprocesses into Horn clauses built over the two predicates $\mathsf{attacker2}$ and $\mathsf{message2}$. For equivalence checking, biprocess is used to represent the pair of processes for which ProVerif is called to check equivalence. The fact $\mathsf{attacker2}(M, M')$ means that the attacker can learn the value M (*resp.* M') from the first (*resp.* second) process encoded by the biprocess. The fact $\mathsf{message2}(M, N, M', N')$ means that the message N (*resp.* N') has appeared on the channel M (*resp.* M') while executing the first (*resp.* second) process encoded by the biprocess.

As for the AKISS tool, our extension of ProVerif consists in the addition of constructs for declaring *everlasting channels* and a *future equational theory* (different from the *present* one). We introduce the extra binary predicate $\mathsf{attacker2\_ev}$ for the generation of Horn clauses from biprocesses of our extended ProVerif language. The fact $\mathsf{attacker2\_ev}(M, M')$ means that in the future, the attacker will either remember or be able to compute from messages he remembers, the value M (*resp.* M'). The declaration of an everlasting channel c generates the following *inheritance* Horn clause:

$$\mathsf{message2} : c[], xm, c[], ym \ \rightarrow\ \mathsf{attacker2\_ev} : xm, ym$$

This clause transports messages sent on the everlasting channel to the “future”. The declaration of future equations generates the same equations as present ones but using our new $\mathsf{attacker2\_ev}$ predicate. For example, the everlasting equation

$$\mathsf{break}(\mathsf{aenc}(\mathsf{pk}(xk), xr, xm)) = xk$$

will generate the two following clauses

$$\begin{array}{l} \mathsf{attacker2\_ev} : x, \mathsf{aenc}(\mathsf{pk}(xk), xr, xm) \ \rightarrow\ \mathsf{attacker2\_ev} : \mathsf{break}(x), xk \\ \mathsf{attacker2\_ev} : \mathsf{aenc}(\mathsf{pk}(xk), xr, xm), x \ \rightarrow\ \mathsf{attacker2\_ev} : xk, \mathsf{break}(x) \end{array}$$

These clauses model the “future” ability of the attacker to recover the decryption key of ciphertexts he remembers.

Using our extension of the ProVerif tool, we managed to find the attack on $\mathsf{Helios}^{\mathsf{noreplay}}$ presented in section 4.2, but also to prove that $\mathsf{Helios}^{\mathsf{noid}}$, $\mathsf{Helios}^{\mathsf{pedersen}}$ and that $\mathsf{Moran-Naor}$ satisfy everlasting vote privacy. However, because of the abstractions made by ProVerif, we had to adapt our models of $\mathsf{Helios}^{\mathsf{noid}}$ and $\mathsf{Helios}^{\mathsf{pedersen}}$ in order for ProVerif to succeed in proving that satisfy everlasting privacy. Indeed, these two protocols do not satisfy uniformity under reductions, and ProVerif reported false attacks on these two protocols. To overcome this limitation of ProVerif, we fixed the order in which the three voters cast their votes.

The tool and example files are available at `http://markryan.eu/research/EverlastingPrivacy/`.

	AKISS	ProVerif
Helios	attack on privacy	attack on privacy
$\mathsf{Helios}^{\mathsf{noreplay}}$	proof of privacy attack on everlasting privacy	proof of privacy attack on everlasting privacy
$\mathsf{Helios}^{\mathsf{noid}}$	proof of everlasting privacy	proof of everlasting privacy (voters casting their votes in fixed order)
$\mathsf{Helios}^{\mathsf{ped}}$	proof of everlasting privacy (2 honest voters only)	proof of everlasting privacy (voters casting their votes in fixed order)
Moran-Naor	proof of everlasting privacy	proof of everlasting privacy

Fig. 3. Automated verification using AKISS and ProVerif.

7 Conclusion

The key idea of "practical" everlasting privacy is that in the future, an attacker will be more powerful in terms of computation (he may be able to break the cryptography) but less powerful in terms of the data he can operate on (transactions between a vote client and the vote server may not have been stored). We realized this idea in the "symbolic" model by allowing different equational theories in different phases, and restricting the information flow from the earlier phase to the later one. We modified ProVerif and AKISS to verify our examples automatically.

We foresee to apply our results to more evolved case studies, e.g. taking into account the zero knowledge proofs that we omitted here for simplicity. Our case studies also show the limitations of the tools for checking equivalence properties which motivates further work to increase their efficiency and scope. Finally, the ability to model different equational theories with restricted information passing between them opens up possibilities for modeling breakable cryptography and other kinds of forward security. In particular it would be interesting to apply the notion of everlasting security to other flavors of anonymity and untraceability.

Acknowledgements. The research leading to these results has received funding from the European Research Council under the European Unions Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement no 258865, project ProSecure, the ANR projects ProSe (decision ANR 2010-VERS-004) and JCJC VIP (decision ANR-11-JS02-006). We also acknowledge funding from EPSRC projects EP/G02684X/1 "Trustworthy Voting Systems" and EP/H005501/1 "Analysing Security and Privacy Properties".

References

1. M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *28th Symposium on Principles of Programming Languages (POPL'01)*. ACM Press, 2001.
2. B. Adida. Helios: web-based open-audit voting. In *17th conference on Security symposium (SS'08)*. USENIX Association, 2008.
3. M. Backes, C. Hritcu, and M. Maffei. Automated verification of remote electronic voting protocols in the applied pi-calculus. In *21st IEEE Computer Security Foundations Symposium (CSF'08)*. IEEE, 2008.
4. B. Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In *14th Computer Security Foundations Workshop (CSFW'01)*. IEEE Comp. Soc. Press, 2001.
5. B. Blanchet, M. Abadi, and C. Fournet. Automated verification of selected equivalences for security protocols. *Journal of Logic and Algebraic Programming*, 75(1), 2008.
6. R. Chadha, Ş. Ciobâcă, and S. Kremer. Automated verification of equivalence properties of cryptographic protocols. In *21th European Symposium on Programming (ESOP'12)*, volume 7211 of *LNCS*. Springer, 2012.
7. D. Chaum, P. Ryan, and S. Schneider. A practical, voter-verifiable election scheme. In *10th European Symposium On Research In Computer Security (ESORICS'05)*, volume 3679 of *LNCS*. Springer, 2005.
8. M. Clarkson, S. Chong, and A. Myers. Civitas: Toward a secure voting system. In *29th IEEE Symposium on Security and Privacy (S&P'08)*, 2008.
9. V. Cortier and B. Smyth. Attacking and fixing helios: An analysis of ballot secrecy. In *24th IEEE Computer Security Foundations Symposium (CSF'11)*, June 2011.
10. E. Cuvelier, T. Peters, and O. Pereira. Election verifiabilty or ballot privacy: Do we need to choose? SecVote, Dagstuhl, 2012. `secvote.uni.lu/slides/opereira-verif-or-priv.pdf`.
11. Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, July 2009.
12. D. Demirel, J. Van De Graaf, and R. Araújo. Improving helios with everlasting privacy towards the public. In *International conference on Electronic Voting Technology/Workshop on Trustworthy Elections (EVT/WOTE'12)*. USENIX Association, 2012.
13. J. Dreier, P. Lafourcade, and Y. Lakhnech. Defining privacy for weighted votes, single and multi-voter coercion. In *17th European Symposium on Research in Computer Security (ESORICS 2012)*, volume 7459 of *LNCS*. Springer, 2012.
14. A. Juels, D. Catalano, and M. Jakobsson. Coercion-resistant electronic elections. In *ACM workshop on Privacy in the electronic society (WPES'05)*. ACM, 2005.
15. S. Kremer and M. D. Ryan. Analysis of an electronic voting protocol in the applied pi-calculus. In *14th European Symposium on Programming (ESOP'05)*, volume 3444 of *LNCS*. Springer, 2005.
16. S. Kremer, M. D. Ryan, and B. Smyth. Election verifiability in electronic voting protocols. In *15th European Symposium on Research in Computer Security (ESORICS'10)*, volume 6345 of *LNCS*. Springer, 2010.
17. R. Küsters, T. Truderung, and A. Vogt. Accountability: definition and relationship to verifiability. In *ACM Conference on Computer and Communications Security (CCS 2010)*, 2010.
18. T. Moran and M. Naor. Receipt-free universally-verifiable voting with everlasting privacy. In *Advances in Cryptology - CRYPTO 2006*, volume 4117 of *LNCS*. Springer, 2006.
19. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology - CRYPTO '91*, volume 576 of *LNCS*. Springer, 1991.