Epistemic Logic for the Applied Pi Calculus*

Rohit Chadha[1], Stéphanie Delaune[2], and Steve Kremer[2]

[1] University of Illinois at Urbana-Champaign, USA
[2] LSV, ENS Cachan & CNRS & INRIA Saclay, France

Abstract. We propose an epistemic logic for the applied pi calculus, which is a variant of the pi calculus with extensions for modeling cryptographic protocols. In such a calculus, the security guarantees are usually stated as *equivalences*. While process calculi provide a natural means to describe the protocols themselves, *epistemic logics* are often better suited for expressing certain security properties such as secrecy and anonymity.
We intend to bridge the gap between these two approaches: using the set of traces generated by a process as models, we define a logic which has constructs for reasoning about both intruder's epistemic knowledge and the set of messages in possession of the intruder. As an example we consider two formalizations of privacy in electronic voting and study the relationship between them.

1 Introduction

The applied pi calculus [2] is an extension of the pi calculus designed for specifying and verifying cryptographic protocols. The main difference from the pi calculus is that it allows one to manipulate complex data, instead of just names. The data is generated by an arbitrary abstract term algebra and interpreted modulo an equational theory. This allows one to abstractly specify cryptographic functions. For instance the equation $\mathsf{dec}(\mathsf{enc}(x, k), k) = x$ models that decryption cancels out encryption if the same key k is used. As the calculus is parametrized by an arbitrary equational theory, several complex cryptographic primitives have been conveniently modeled in literature. For example, blind signatures were modeled in [13] and non-interactive zero-knowledge proofs were modeled in [3]. This calculus has been successfully used to study a variety of security protocols, e.g. the direct anonymous attestation protocol [3], some electronic voting protocols [13]. Moreover, there exists tool support [5] for assisting the verification of protocols in the applied pi calculus.

As argued above the applied pi calculus is a convenient and flexible formalism for describing the processes which model the protocol. However, security properties are more difficult to specify. Some properties may directly be specified using observational equivalence, but this is generally not very natural and convenient. A more natural approach to verify protocols for correctness would be to define a suitable logic interpreted over the terms of the calculus and express the desired security goal in that logic.

Our main contribution is the definition of an epistemic logic for the applied pi calculus suitable for expressing important security goals. The logic itself is an LTL like

* This work has been partially supported by the ANR project SeSur AVOTÉ

temporal logic with a special predicate Has that models deducibility of messages by an intruder and an epistemic knowledge operator K which allows us to reason about the intruder's *epistemic knowledge*. Other predicates of the logic are defined by *events* which annotate the protocol. Similar annotations have already been used for specifying authentication properties, initially by Woo and Lam [20] and more specifically in the applied pi calculus by Blanchet [6]. We emphasize here that our main motivation behind designing this logic is to express important security goals and *not* to study observational equivalence. In particular, a Hennessy-Milner theorem will not hold: observationally equivalent processes may satisfy different security goals.

Epistemic logics, going back to the BAN logic [8], are well-suited to express complex security properties. At that time, the logic was used to reason about authentication protocols. However, epistemic knowledge is particularly useful when reasoning about anonymity properties (*e.g.*, see [18]). Intuitively, an intruder (epistemically) knows that a property ϕ is true, if ϕ is true on every run which is indistinguishable for the intruder from the current one. In general epistemic logics this is modeled by an arbitrary equivalence relation on runs. In the context of security protocols, equivalence of runs is tightly related to the cryptographic functions used: an intruder which does not know k, should regard the runs outputting respectively $\mathsf{enc}(0, k)$ and $\mathsf{enc}(1, k)$ as equivalent. We formalize equivalence of runs by lifting the notion of *static equivalence* to protocol runs. We emphasize here that our logic contains the epistemic modality *only* for the intruder and not for other participants. This is primarily because the processes only keep track of messages in possession of the intruder.

We illustrate the expressiveness of our logic by expressing a range of security properties: secrecy, authentication as well as fairness in contract signing protocols. We then specify *privacy* in voting protocols, which relies on the epistemic knowledge of the intruder. We show that a definition of vote privacy in terms of process equivalence as defined in [13] implies vote privacy in terms of epistemic logic, as defined in [4]. Then we slightly weaken the equivalence based definition, replacing observational equivalence with trace equivalence. In that case, under reasonable assumptions, we show that the converse implication, i.e. epistemic privacy implies privacy as equivalence, also holds. This result is important in that it clarifies the relationship between two definitions of privacy employed in the literature. Furthermore, the result suggests that trace equivalence is more appropriate to model voter privacy even though observational equivalence is convenient to use because of the available tool support.

For the rest of the paper we reserve the phrase "intruder's knowledge" for his epistemic knowledge. We use the word "intruder's possession" for the set of messages that an intruder possesses (which is sometimes referred to as knowledge in security).

2 The applied pi calculus

We present here the syntax and semantics of a slightly enriched applied pi calculus [2].

2.1 Syntax

The syntax of the applied pi-calculus assumes an order-sorted vocabulary consisting of a denumerable set of *names* of each sort, a denumerable set of *variables* of each sort and a *signature* Σ consisting of a finite set of *function symbols* with their arity. The details of the sort system are unimportant, as long as it differs *base types* and *channel types*. We always suppose that function symbols only operate on and return terms of base type. The grammar of the set of terms is defined as:

$$\begin{array}{lll} M, N, T := & & \text{terms} \\ & a, b, \ldots, \ldots k, m, n, \ldots & \text{names} \\ & x, y, z, \ldots & \text{variables} \\ & f(M_1, M_2, \ldots M_k) & \text{function application} \end{array}$$

Of course function symbol application must respect sorts and arities. We shall use $u, v, \ldots$ to range over both names and variables. We write $vars(T)$ for the set of variables occurring in T. T is said to be a *ground* term if $vars(T) = \emptyset$.

Example 1. Let $\Sigma = \{\mathsf{enc}/2, \mathsf{dec}/2, \mathsf{pair}/2, \mathsf{proj}_1/1, \mathsf{proj}_2/1\}$ be a signature containing function symbols for encryption, decryption and pairing, each of arity 2, as well as left and right projection symbols of arity 1. The term $\mathsf{enc}(a, k)$ is ground.

There are two kinds of processes in the applied pi calculus– *plain* processes built up in a similar way to processes in the pi calculus except that messages can contain terms rather than just names, and *extended* processes which add *active substitutions* (explained below) and restriction on variables. Furthermore, we enrich plain processes with non-deterministic choice and a set of events $\mathrm{e}, \mathrm{e}_1, \ldots$ (parametrized by a sequence of terms of the correct sort). These events are "annotations" which are useful in formalizing security properties and (as we shall see later) play no part in observational equivalence. Extended processes are also enriched with *event stores*, which record the events that happen along an execution. We do not have replication in our calculus.

$$\begin{array}{lll} P, Q, R := & & \text{plain processes} \\ & 0 & \text{null process} \\ & P \mid Q & \text{parallel composition} \\ & P + Q & \text{non-det. choice} \\ & \nu n.P & \text{name restriction} \\ & \text{if } M = N \text{ then } P \text{ else } Q & \text{conditional} \\ & \text{in}(u, x).P & \text{message input} \\ & \text{out}(u, N).P & \text{message output} \\ & \mathrm{e}(\widetilde{M}).P & \text{event} \end{array} \qquad \begin{array}{lll} A, B, C := & & \text{extended processes} \\ & P & \text{plain process} \\ & A \mid B & \text{parallel composition} \\ & \nu n.A & \text{name restriction} \\ & \nu x.A & \text{variable restriction} \\ & \{^M/_x\} & \text{active substitution} \\ & [\mathrm{e}(\widetilde{M})] & \text{event store} \end{array}$$

$\{^M/_x\}$ is the active substitution that replaces the variable x with the term M. Active substitutions generalize the "let" construct: $\nu x.(\{^M/_x\} \mid P)$ corresponds exactly to "let $x = M$ in P". An event store $[\mathrm{e}(\widetilde{M})]$ memorizes that the event $\mathrm{e}(\widetilde{M})$ happened. As usual, names and variables have scopes, which are delimited by restrictions and by inputs. Please note that the "event" construct is not a binding construct. We write $fv(A)$, $bv(A)$, $fn(A)$ and $bn(A)$ for the sets of *free* and *bound variables* and *free* and *bound names* of A, respectively. We say that an extended process is *closed* if all its variables are either bound or defined by an active substitution. An *evaluation context* $C[\_]$ is an extended process with a hole instead of an extended process.

Active substitutions are useful because they allow us to map an extended process A to its *frame*, denoted $fr(A)$, by replacing every plain process and event store in A with 0.

A frame is an extended process built up from 0 and active substitutions by parallel composition and restriction. The frame $fr(A)$ accounts for the set of terms statically possessed by the intruder (but does not account for A's dynamic behavior). The *domain of a frame* φ, $\mathrm{dom}(\varphi)$, is the set of variables for which φ defines a substitution (i.e. variables x for which φ contains a substitution $\{^M/_x\}$ not under a restriction on x). In such a case, i.e. when $x \in \mathrm{dom}(\varphi)$, x allows the intruder to refer to the term M.

2.2 Semantics

The semantics is defined in terms of a LTS which records the interaction of an extended process with the intruder. We associate an equational theory E to the signature Σ. E is defined by a set of equations $M = N$ and induces an equivalence relation over terms: $=_\mathsf{E}$ is the smallest equivalence relation on terms, which contain all equations in E and is closed under substitution of terms for variables and bijective renaming of names.

Example 2. Considering the signature Σ of Example 1 we define the equational theory $\mathsf{E_{enc}}$ by the equations $\mathsf{dec}(\mathsf{enc}(x,y),y) = x$ and $\mathsf{proj}_i(x_1,x_2) = x_i$ for $i \in \{1,2\}$. We have that $\mathsf{dec}(\mathsf{enc}(a,k),k) =_{\mathsf{E_{enc}}} a$.

We define the relation $\cong$ to be the smallest equivalence relation on extended processes that is closed under application of evaluation contexts and such that

PAR-0	$A \mid 0 \cong A$	CHOICE-A	$P + (Q + R) \cong (P + Q) + R$
PAR-A	$A \mid (B \mid C) \cong (A \mid B) \mid C$	CHOICE-C	$P + Q \cong Q + R$
PAR-C	$A \mid B \cong B \mid A$	ALIAS	$\nu x.\{^M/_x\} \cong 0$
NEW-C	$\nu u.\nu v.A \cong \nu v.\nu u.A$	SUBST	$\{^M/_x\} \mid A \cong \{^M/_x\} \mid A\{^M/_x\}$
NEW-PAR	$A \mid \nu u.B \cong \nu u.(A \mid B)$ if $u \notin fn(A) \cup fv(A)$	REWRITE	$\{^M/_x\} \cong \{^N/_x\}$ if $M =_\mathsf{E} N$

We define *structural equivalence*, $\equiv$, to be $\cong$ closed under α-conversion on names and variables. In comparison to the original applied pi calculus we dropped the structural equivalence $\nu n.0 \equiv 0$ which will be important for deduction.

Example 3. Consider the following process P:

$$\nu s,k.(\mathsf{out}(c_1,\mathsf{enc}(s,k)) \mid \mathsf{in}(c_1,y).\mathsf{out}(c_2,\mathsf{dec}(y,k))).$$

The first component publishes the message $\mathsf{enc}(s,k)$ by sending it on c_1. The second receives a message on c_1, uses the secret key k to decrypt it, and forwards the resulting plaintext on c_2. P is structurally equivalent to the following extended process A:

$$A = \nu s,k,x_1.(\mathsf{out}(c_1,x_1) \mid \mathsf{in}(c_1,y).\mathsf{out}(c_2,\mathsf{dec}(y,k)) \mid \{^{\mathsf{enc}(s,k)}/_{x_1}\})$$

We have $fr(A) = \nu s,k,x_1.\{^{\mathsf{enc}(s,k)}/_{x_1}\} \cong \nu s,k.0$ (since x_1 is under a restriction).

Internal reduction $\rightarrow$ is the smallest relation on extended processes closed under structural equivalence and application of evaluation contexts such that

COMM	$\mathsf{out}(a,M).P \mid \mathsf{in}(a,x).Q \rightarrow P \mid Q\{^M/_x\}$	THEN	if $M = N$ then P else $Q \rightarrow P$ where $M =_\mathsf{E} N$
EVENT	$\mathsf{e}(\widetilde{M}).P \rightarrow P \mid [\mathsf{e}(\widetilde{M})]$	ELSE	if $M = N$ then P else $Q \rightarrow Q$ where M, N are ground and $M \neq_\mathsf{E} N$.
CHOICE	$P + Q \rightarrow P$		

As usual $\to^*$ denotes the reflexive transitive closure of $\to$.

The operational semantics is extended by a *labeled* operational semantics enabling us to reason about processes that interact with their environment. Below, a and c are channel names, x is a variable of base type and y is a variable of any type.

IN $\quad \mathrm{in}(a,y).P \xrightarrow{in(a,M)} P\{^M/_y\}$

OUT-CH $\quad \mathrm{out}(a,c).P \xrightarrow{out(a,c)} P$

OPEN-CH $\quad \dfrac{A \xrightarrow{out(a,c)} A' \qquad c \neq a}{\nu c.A \xrightarrow{\nu c.out(a,c)} A'}$

OUT-T $\quad \mathrm{out}(a,M).P \xrightarrow{\nu x.out(a,x)} P \mid \{^M/_x\}$ $\quad x \notin fv(P) \cup fv(M)$

SCOPE $\quad \dfrac{A \xrightarrow{\ell} A' \quad u \text{ does not occur in } \ell}{\nu u.A \xrightarrow{\ell} \nu u.A'}$

PAR $\quad \dfrac{A \xrightarrow{\ell} A' \quad \begin{array}{c} bn(\ell) \cap fn(B) = \emptyset \\ bv(\ell) \cap fv(B) = \emptyset \end{array}}{A \mid B \xrightarrow{\ell} A' \mid B}$

STRUCT $\quad \dfrac{A \equiv B \quad B \xrightarrow{\ell} B' \quad A' \equiv B'}{A \xrightarrow{\ell} A'}$

Example 4. Continuing Example 3, we have that

$$A \xrightarrow{\nu x_1.out(c_1,x_1)} \xrightarrow{in(c_1,x_1)} \nu s,k.(\mathrm{out}(c_2,s) \mid \{^{\mathsf{enc}(s,k)}/_{x_1}\}) \stackrel{\mathsf{def}}{=} A'.$$

The frame associated to A' is $fr(A') = \nu s,k.\{^{\mathsf{enc}(s,k)}/_{x_1}\}$.

2.3 Equivalences

In this section we introduce two notions of process equivalences: *trace equivalence* and *labeled bisimulation*. These definitions are based on *static equivalence*, an equivalence on frames, and *static equivalence of traces*, which lifts static equivalence from frames to traces. Static equivalence is a notion of intruder's possession that has been extensively studied (e.g. [1]). Another notion, namely deducibility will be discussed in Section 3. The notion of static equivalence is useful to define labeled bisimilarity.

Definition 1 (static equivalence). *We say that two terms M and N are* equal in the frame *ϕ, and write $(M =_{\mathsf{E}} N)\phi$, if there exists $\tilde{n}$ and a substitution σ such that $\phi \equiv \nu\tilde{n}.\sigma$, $\tilde{n} \cap (fn(M) \cup fn(N)) = \emptyset$, and $M\sigma =_{\mathsf{E}} N\sigma$. We say that two closed frames ϕ_1 and ϕ_2 are* statically equivalent*, $\phi_1 \sim \phi_2$, when:*

- $\mathrm{dom}(\phi_1) = \mathrm{dom}(\phi_2)$*, and*
- *for all terms M, N we have that $(M =_{\mathsf{E}} N)\phi_1$ if and only if $(M =_{\mathsf{E}} N)\phi_2$.*

Example 5. Let $\phi = \nu k,s.(\{^{\mathsf{enc}(s,k)}/_{x_1}\} \mid \{^k/_{x_2}\})$, $\phi' = \nu k.(\{^{\mathsf{enc}(s',k)}/_{x_1}\} \mid \{^k/_{x_2}\})$ where s, s', k are names. We have $(\mathsf{dec}(x_1,x_2) =_{\mathsf{E_{enc}}} s')\phi'$ but $(\mathsf{dec}(x_1,x_2) \neq_{\mathsf{E_{enc}}} s')\phi$, thus $\phi \not\sim \phi'$ (for $\mathsf{E_{enc}}$). However, $\nu k,s.\{\mathsf{enc}(s,k)/x_1\} \sim \nu k.\{\mathsf{enc}(s',k)/x_1\}$.

We now define two notions of indistinguishability in the presence of an active intruder. The first one is *trace equivalence*, the second one *labeled bisimulation*. As we are interested in the interactions of a process with the intruder (and not just the internal actions), we use the labeled transition system to define the possible "runs" of a process:

Definition 2 (trace). *A* trace tr *is a finite derivation* $\mathsf{tr} = A_0 \xrightarrow{\ell_1} A_1 \ldots \xrightarrow{\ell_n} A_n$ *such that each A_i is a closed extended process where each ℓ_i is either empty (and represents* *an internal action) or is a labeled action ℓ_i with $fv(\ell_{i+1}) \subseteq \mathrm{dom}(A_i)$. The trace* tr *is said to be maximal if $A_n \stackrel{\ell}{\not\rightarrow}$ for any ℓ.*

We write $\mathsf{tr}[i]$ for the process A_i and $\mathsf{tr}[i,j]$ for the trace $A_i \xrightarrow{\ell_{i+1}} A_{i+1} \ldots \xrightarrow{\ell_j} A_j$. We shall say that $|\mathsf{tr}| = n$.

We say that the trace tr *is of the form $A_0 \rightarrow^* \xrightarrow{\ell_{i_1}} A_{i_1} \rightarrow^* \xrightarrow{\ell_{i_2}} A_{i_{j+1}} \ldots \rightarrow^* \xrightarrow{\ell_{i_r}} A_r \rightarrow^* A_n$ if ℓ_k is a labeled action for all $k = i_j, 1 \leq j \leq r$ and the internal action otherwise.*

Given a process A we define $\mathsf{tr}(A)$ to be the set of all traces tr *such that $\mathsf{tr}[0] = A$ and $\mathsf{tr}_{\mathsf{max}}(A)$ to be the set of all the maximal traces* tr *such that $\mathsf{tr}[0] = A$.*

In order to define trace equivalence we lift static equivalence from frames to traces. In order to ensure that bisimilar processes are also trace equivalent we need to define α-equivalence of traces. Intuitively, we say that a labeled action ℓ in a trace tr binds n in the subsequent trace if n occurs as a bound name in ℓ. A trace tr can be α-renamed to tr' if tr' can be obtained by an α-renaming of the bound name n. The formal definition has been moved to the Appendix where its motivation is also discussed. We write $\mathsf{tr} \rightarrow_\alpha \mathsf{tr}'$ if tr' is obtained from tr by an α-renaming of a bound name. The relation $\sim_\alpha$ is defined to be the reflexive, symmetric and transitive closure of $\rightarrow_\alpha$.

Intuitively, we say that two traces are statically equivalent to the intruder if the intruder performed the same actions in the trace and the intruder could not "statically" distinguish the processes resulting from these actions. Formally,

Definition 3 (static equivalence of traces ($\sim_t$)). *Let* tr *be a trace of the form $A_0 \rightarrow^* \xrightarrow{\ell_1} A_1 \rightarrow^* \xrightarrow{\ell_2} A_{j+1} \ldots \rightarrow^* \xrightarrow{\ell_r} A_r \rightarrow^* B$. Let tr' be a trace of the form $A'_0 \rightarrow^* \xrightarrow{\ell'_1} A'_1 \rightarrow^* \xrightarrow{\ell'_2} A'_{j+1} \ldots \rightarrow^* \xrightarrow{\ell'_l} A'_l \rightarrow^* B'$. Then $\mathsf{tr} \leftrightarrow_t \mathsf{tr}'$ if $r = l$, and*

- *for all $1 \leq i \leq r$, $\ell_i = \ell'_i$.*
- *for all $0 \leq i \leq r$, $fr(A_i) \sim fr(A'_i)$ (static equivalence).*

The relation $\sim_t$ is the transitive closure of $\sim_\alpha \cup \leftrightarrow_t$.

We can now define trace equivalence.

Definition 4 (trace equivalence ($\approx_t$)). *Let A and B be two closed extended processes. We say that A is* trace included *in B, written $A \subseteq_t B$ if for each trace $\mathsf{tr}_A \in \mathsf{tr}(A)$ there exists $\mathsf{tr}_B \in \mathsf{tr}(B)$ such that $\mathsf{tr}_A \sim_t \mathsf{tr}_B$. The processes A and B are* trace equivalent*, written $A \approx_t B$, if $A \subseteq_t B$ and $B \subseteq_t A$.*

Trace equivalence is an appealing notion for modeling indistinguishability in presence of an active intruder and can be used to formalize many security properties (e.g. strong secrecy, anonymity properties, ...). However, bisimulation is often considered as it has better proof techniques and is easier to manipulate.

Definition 5 (labeled bisimilarity ($\approx$)). Labeled bisimilarity *is the largest symmetric relation $\mathcal{R}$ on closed extended processes, such that $A \mathcal{R} B$ implies*

1. *$fr(A) \sim fr(B)$;*

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

As expected labeled bisimulation implies trace equivalence, i.e. $\approx \subset \approx_t$. Hence bisimulation can be used as a proof technique to show trace equivalence.

3 Epistemic Logic

We shall now present the epistemic logic which allows us to reason about intruder's epistemic knowledge and the set of facts in its possession.

3.1 Syntax

The formulas of our logic consist of two levels. *Static formulas* are used to reason about a "snapshot" of the process. They include predicates for events that may have occurred in the past and a predicate for a set of terms that the intruder statically possesses. *Epistemic formulas* allow us to reason about the dynamic behavior of the process and the epistemic knowledge that the intruder can deduce from its past interactions with the process. The formulas use a term language which denotes the set of messages. The syntax of the logic is given in BNF form in Table 1 and discussed below.

Terms.
$\widehat{T} ::= \widehat{n} \;[\!]\; z \;[\!]\; \widehat{f}(\widehat{T}, \ldots, \widehat{T})$

Static formulas.
$\delta ::= \top \;[\!]\; \mathsf{Has}(\widehat{T}) \;[\!]\; \widehat{\mathsf{evt}}(\widehat{T}, \ldots, \widehat{T}) \;[\!]\; \neg\delta \;[\!]\; \delta \vee \delta \;[\!]\; \exists z.\delta$

Epistemic formulas (with the proviso δ is a closed formula and has no free names).
$\phi ::= \delta \;[\!]\; \neg\phi \;[\!]\; \phi \vee \phi \;[\!]\; \mathsf{K}\phi \;[\!]\; \Box\phi \;[\!]\; \boxminus\phi$

Table 1. Syntax of the Epistemic Logic

Term language. For the term language of our logic we shall assume that for each name n in the vocabulary of the applied pi calculus, there is a unique name $\widehat{n}$ in the logic. Similarly for each function symbol f in the vocabulary of the applied pi calculus, we have a unique function symbol $\widehat{f}$ in the logic. However, there is no particular correspondence between the set of variables in the logic and the applied pi calculus. We use $z, z_1, \ldots$ to range over the variables of the logic. The set of terms of the logic now consist of names, variables and function application (the usual restriction on sorts and arity apply here).

Static formulas. Static formulas assume a unary predicate Has whose argument is of base sort. This predicate is used to reason about the set of terms that the intruder possesses. It also assumes that for each event evt in the set of events for the calculus there is predicate $\widehat{\mathsf{evt}}$ (of the correct sort and arity). These predicates are used to reason about events that may have occurred in the past. The static formulas are built from these predicates using the connectives $\top$, negation $\neg$, disjunction $\vee$ and existential quantification $\exists z$. The usual connectives $\wedge$ $\perp$ and $\Rightarrow$ and the universal quantification $\forall$ can be derived from these connectives. We also assume the standard definitions of free and bound variables and substitution. A static formula is *closed* if it does not contain any free variable.

Epistemic formulas. Epistemic formulas reason about dynamic behavior of a process and are constructed from *closed* static formulas with no free names using the connectives conjunction $\wedge$, negation $\neg$, disjunction $\vee$, existential quantification $\exists z$ and the modalities $\Box, \boxminus, \mathsf{K}$. The reason for using only closed formulas will become clear in Section 3.2. Disallowing names is not restrictive, as events can be used to refer to names. The formulas are interpreted over the possible "runs" of the process. The formula $\Box\phi$ is true at some point in a run if ϕ is true for all possible future points whereas the formula $\boxminus\phi$ is true if ϕ is true for all past points. The formula $\mathsf{K}\phi$ is true if the intruder knows (in the epistemic sense) ϕ to be true based upon its interaction with the process in the past. The connectives $\perp$ and $\Rightarrow$ and the modality $\Diamond$ can be derived.

3.2 Semantics

We now define the semantics of the logic. We start by the denotation of terms.

Denotation of Terms. The terms of the logic are interpreted as ground terms of the applied pi calculus and use the concept of an assignment. An *assignment* ρ is a map which maps each logic variable $z \in \mathcal{Z}$ to a ground term of the applied pi calculus. Using the assignment ρ, the denotation of terms is defined inductively as

$$[\![\widehat{n}]\!]_\rho = n \qquad [\![\widehat{z}]\!]_\rho = \rho(z) \qquad [\![\widehat{f}(\widehat{T_1}, \ldots, \widehat{T_r})]\!]_\rho = f([\![\widehat{T_1}]\!]_\rho, \ldots, ([\![\widehat{T_r}]\!]_\rho)$$

Satisfaction of static formulas. The models of static formulas are pairs- one part of which is a *name distinct* closed extended process A term, i.e. a process such that $bn(A) \cap fn(A) = \emptyset$ and no name is bound twice; and the other part an assignment.

We need another definition for our semantics which formalizes a second notion of intruder's possession (e.g. [1]).

Definition 6 (Deducibility). *Let $\phi \cong \nu\tilde{n}.\sigma$ be a closed name-distinct frame and M be a term. We say that M is* deducible *from ϕ, denoted by $\phi \vdash M$ if there exists a term N such that $fn(N) \cap \tilde{n} = \emptyset$ and $N\sigma =_{\mathsf{E}} M$. Such a term N is a* recipe *of the term M.*

Note that when $\nu\tilde{n}.\sigma \vdash M$, any occurrence of names from $\tilde{n}$ in M is bound by $\nu\tilde{n}$. It is for this reason that we introduce the relation $\cong$ (*cf.* Example in Remark 1, item 3).

Example 6. Consider the two frames ϕ and ϕ' given in Example 5. We have that $\phi \vdash k$, $\phi \vdash s$ and $\phi \vdash s'$. Indeed x_2, $\mathsf{dec}(x_1, x_2)$ and s' are recipes of the terms k, s and s'.

The interpretation of the static formulas given a name-distinct process term A and an assignment ρ is defined in Table 2. The interesting cases are the satisfaction of the predicates Has and $\widehat{\mathsf{evt}}$. Intuitively, the formula $\mathsf{Has}(\widehat{T})$ is satisfied if the intruder can deduce the denotation of $\widehat{T}$. The formula $\widehat{\mathsf{evt}}(\widehat{T}_1, \ldots, \widehat{T}_r)$ is satisfied if the corresponding event $\mathsf{evt}([\![\widehat{T}_1]\!]_\rho, \ldots, [\![\widehat{T}_r]\!]_\rho)$ has occurred. The other definitions are standard. Note that the assignment $\rho[z \mapsto M]$ is the same as ρ except that on z it takes the value M and the formula $\delta[M/z]$ is the formula obtained from δ by substituting the free occurrences of z by M.

Remark 1.

1. If the formula δ is closed, *i.e.*, does not contain any free variables, then the satisfaction of δ depends only on the process and is independent of the assignment. For such formulas we can drop the assignment in the satisfaction relation.

$A, \rho \models \top$		always
$A, \rho \models \widehat{\mathsf{evt}}(\widehat{T}_1, \ldots, \widehat{T}_r)$	*iff*	$A \cong \nu\tilde{n}.(A \mid [\mathsf{evt}(M_1, \ldots, M_r)]) \wedge M_i =_{\mathsf{E}} [\![\widehat{T}_i]\!]_\rho \;\; 1 \leq i \leq r$
$A, \rho \models \mathsf{Has}(\widehat{T})$	*iff*	$fr(A) \vdash [\![\widehat{T}]\!]_\rho$
$A, \rho \models \neg\delta$	*iff*	$A, \rho \not\models \delta$
$A, \rho \models \delta_1 \vee \delta_2$	*iff*	$A, \rho \models \delta_1$ or $A, \rho \models \delta_2$
$A, \rho \models \exists z.\delta$	*iff*	$\exists$ a ground term M such that $A, \rho[z \mapsto M] \models \delta[M/z]$

Table 2. Satisfaction of static formulas

2. Note that name-distinctness is crucial for the definition of satisfaction of the static formulas. The name distinctness allows us to uniquely identify the bound names and interpret them. Otherwise, the process $A = (\nu n.[\mathsf{evt}_1(n)]) \mid (\nu n.[\mathsf{evt}_2(n)])$ will satisfy $\widehat{\mathsf{evt}}_1(\widehat{n}) \wedge \widehat{\mathsf{evt}}_2(\widehat{n})$ which is clearly wrong as the two bound names refer to different nonces.
3. For a similar reason, we need to forbid α-renaming when evaluating predicates evt. Otherwise, (if we replace $\cong$ with $\equiv$ in the above semantics) we have that
$$\nu n_1, n_2.([\mathsf{evt}_1(n_1)] \mid [\mathsf{evt}_2(n_2)]) \models \exists z.\ (\widehat{\mathsf{evt}_1}(z) \wedge \widehat{\mathsf{evt}_2}(z)).$$
4. It can be checked that for any name-distinct closed frame ϕ, if $\phi \cong \nu\tilde{n}.\sigma$ and $\phi \cong \nu\tilde{n'}.\sigma'$ then $\tilde{n}$ and $\tilde{n'}$ are the same (upto ordering) and for any N such that $fn(N) \cap \tilde{n} = \emptyset$, $N\sigma =_{\mathsf{E}} N\sigma'$. Hence, if $A_1 \cong A_2$, we get that A_1 and A_2 satisfy the same set of static formulas.
5. The previous observation would not have been true if we had allowed the equivalence $\nu n.0 \equiv 0$. In particular, the intruder can deduce all ground terms in the process 0 while it cannot deduce the term n in the process $\nu n.0$.

Please note that even name-distinct processes which are equal modulo α-conversion may satisfy different static formula. However, if we limit ourselves to closed formulas with no free names, α-renaming does not affect the satisfaction.

Lemma 1. *Let δ be a closed static formula with no free names and A_1 and A_2 be two name distinct extended processes such that $A_1 \equiv A_2$. Then $A_1 \models \delta$ iff $A_2 \models \delta$.*

The above Lemma allows us to define the semantics of the epistemic formulas.

Satisfaction of epistemic formulas. We shall now define the satisfaction relation for epistemic formulas. As in the case of epistemic logic for distributed systems [14, 15], the epistemic formulas will be interpreted over the possible "runs" of a process, i.e. the set of maximal traces (Definition 2). Please note that since we do not have replication in our process terms, all traces of a process are finite and our definition of maximal traces does capture all possible "runs". The traces are enough to interpret the temporal modalities $\Box$ and $\boxminus$. In order to interpret the modality K, we need to consider an equivalence relation on the set of traces which identifies traces that are indistinguishable to the intruder: static equivalence on traces (Definition 3). An epistemic formula ϕ is interpreted over a triple - a closed extended process A, a maximal trace $\mathsf{tr} \in \mathsf{tr}_{\mathsf{max}}(A)$ and a position $0 \leq j \leq |\mathsf{tr}|$ in tr as described in Table 3.

Remark 2. Our use of static equivalent traces as indistinguishable traces is reminiscent of what is often called *perfect recall* in distributed systems- the intruder distinguishes

$A, \mathsf{tr}, i \models \delta$	iff there is a name-distinct extended process A' such that $\mathsf{tr}[i] \equiv A'$ and $A' \models \delta$
$A, \mathsf{tr}, i \models \Box\phi$	iff $\forall i \leq j \leq \lvert\mathsf{tr}\rvert.\ A, \mathsf{tr}, j \models \phi$
$A, \mathsf{tr}, i \models \boxminus\phi$	iff $\forall 0 \leq j \leq i.\ A, \mathsf{tr}, j \models \phi$
$A, \mathsf{tr}, i \models \mathsf{K}\phi$	iff $\forall \mathsf{tr}' \in \mathsf{tr}_{\mathsf{max}}(A), \forall 0 \leq j \leq \lvert\mathsf{tr}'\rvert$ such that $\mathsf{tr}[0,i] \sim_t \mathsf{tr}'[0,j] \Rightarrow A, \mathsf{tr}', j \models \phi$
$A, \mathsf{tr}, i \models \neg\phi$	iff $A, \mathsf{tr}, i \not\models \phi$
$A, \mathsf{tr}, i \models \phi_1 \vee \phi_2$	iff $A, \mathsf{tr}, i \models \phi_1$ or $A, \mathsf{tr}, i \models \phi_2$

Table 3. Satisfaction of epistemic formulas

traces based upon the complete history of its interaction with the process. We could have, of course, chosen to define coarser equivalence relations. For example, we could have declared two traces to be equivalent if the intruder cannot "statically" distinguish the last processes in the respective traces.[3] However, a coarser relation would result in intruder "knowing" a smaller set of formulas to be true which may lead to declaring a protocol secure which otherwise will be insecure. Besides, an all powerful intruder should be able to record its history of interaction with the protocol.

Definition 7. *We say that $A \models \phi$ if for all $\mathsf{tr} \in \mathsf{tr}_{\mathsf{max}}(A)$ we have $A, \mathsf{tr}, 0 \models \phi$.*

Not that Lemma 1 will not be true if we replace structural equivalence with static equivalence. One reason is the presence of the predicates $\widehat{\mathsf{evt}}$ as static equivalence does not depend on presence/absence of such formulas. However, even if we were to consider the fragment of the logic without these predicates, statically equivalent processes may satisfy different static formulas (and thus Hennessy-Milner Theorem does not hold).

Lemma 2. *There are closed extended processes A_1 and A_2 and an epistemic formula ϕ such that $A_1 \approx A_2$ and $A_1 \models \phi$ but $A_2 \not\models \phi$.*

Proof. Consider the two processes $A_1 = \nu n.\{^{\mathsf{hash}(n)}/_x\}$ and $A_2 = \nu n.\{^{n}/_x\}$ where hash is unary function symbol which models a cryptographic hash function and hence cannot be inverted. We assume that the set of equations E is empty. We have that $A_1 \approx A_2$. We have also that $A_1 \models \exists z.(\mathsf{Has}(\mathsf{hash}(z)) \wedge \neg\mathsf{Has}(z))$ (the intruder has the hash of the nonce n but cannot invert it) while $A_2 \not\models \exists z.(\mathsf{Has}(\mathsf{hash}(z)) \wedge \neg\mathsf{Has}(z))$ (the intruder has every free name and can create its hash). $\Box$

3.3 Examples

We now give some simple examples of security protocols that can be modeled in our logic. These examples do not use the knowledge operator. We refer to Section 4 for such an example. We only consider closed formulas (no free variables) and formulas without names. The idea is to annotate the process and to use the parametric events to refer to bound names. Specifically, we will show how to specify secrecy, authentication and fairness in exchange protocols in our formalism.

[3] This is similar in spirit to what is commonly called "knowledge" in security.

Example 7. This is a way to express the secret (in the sense of deducibility) of the name s in $P = \nu s.\mathsf{evt}(s).\mathsf{out}(c,s)$. Let $\phi = \Box\forall z.(\mathsf{evt}(z) \Rightarrow \neg\mathsf{Has}(z))$. Obviously, we have $P \not\models \phi$ as $P \to A_1 \xrightarrow{\nu x.out(c,x)} A_2$ is a trace in $\mathsf{tr}_{\mathsf{max}}(P)$ where $A_1 = \nu s.(\mathsf{out}(c,s) \mid [\mathsf{evt}(s)])$, $A_2 = \nu s.(\{^s/_x\} \mid [\mathsf{evt}(s)])$ and $(P, \mathsf{tr}, 2) \models \mathsf{evt}(s) \wedge \mathsf{Has}(s)$.

Another classical example is authentication modeled as an agreement property.

Example 8. Consider the following simple handshake protocol where k is a shared key and f any free symbol:

$$\begin{array}{l} \mathsf{A} \ \rightarrow \ \mathsf{B} : \mathsf{enc}(n,k) \\ \mathsf{B} \ \rightarrow \ \mathsf{A} : \mathsf{enc}(f(n),k) \end{array}$$

The goal of this protocol is to authenticate B from A's point of view. In the applied pi calculus this protocol is modeled by $\nu k.(A \mid B)$ where

$$\begin{array}{l} A = \nu n.\ \mathsf{out}(\mathsf{enc}(n,k)).\ \mathsf{in}(x).\ \mathsf{if}\ \mathsf{dec}(x,k) = f(n)\ \mathsf{then}\ \mathsf{end}(n) \\ B = \mathsf{in}(y).\ \mathsf{begin}(\mathsf{dec}(y,k)).\ \mathsf{out}(\mathsf{enc}(f(\mathsf{dec}(y,k)),k)) \end{array}$$

The events begin and end are used to annotate the protocol. The authentication of B to A is then modeled by $\phi = \Box\forall z.(\mathsf{end}(z) \Rightarrow \mathsf{begin}(z))$.

Yet another, less classical example of property is fairness in contract signing protocols.

Example 9. In a fair contract signing protocols two agents want to exchange their corresponding signatures on a given contract in such a way that at the end of the protocol either both participants obtain the signed contract or none of them does so. Describing a complete example of such a protocol would be out of the scope of this paper and we refer the reader to [9] for more details. These protocols either terminate in a final state where the exchange has been aborted or in a final state where the exchange did succeed. For the purpose of our example, we suppose that the process modeling the participant P (either A or B) is annotated as follows: the event $\mathsf{Pend}(c)$ indicates that P is in a final state for some contract c; the event $\mathsf{Pcontract}(c)$ indicates that P successfully received the signed contract. Then, *fairness for A* can be modeled as

$$\phi = \Box\forall c.(\mathsf{Aend}(c) \Rightarrow (\neg\mathsf{Bcontract}(c) \vee \mathsf{Acontract}(c))).$$

The formula says that for any contract whenever A is in a final state ($\mathsf{Aend}(c)$), either B did not obtain the contract signed by A ($\neg\mathsf{Bcontract}(c)$) or A did obtain the contract signed by B ($\mathsf{Acontract}(c)$). Fairness for B can be modeled in a similar way.

4 Privacy in electronic voting protocols

Many electronic voting protocols have been proposed in the literature and their formal analysis has received considerable attention [13, 4]. One important security goal is *privacy of votes*– an intruder should not be able to learn (by its interaction with the protocol) how an honest voter Alice voted. This property has been formulated both as an observational equivalence, e.g. in [13], and as an epistemic property, e.g. in [4], although never within the same formalism. Our formalism allows us to consider both the formalizations and compare them within the same framework. For the sake of simplicity, we only consider single protocol instances in which two voters Alice and Bob participate and we assume that there are only two voting options available to Alice and Bob and we represent these options by $\mathbf{0}$ and $\mathbf{1}$.

Electronic voting protocols in applied pi calculus. We refer the reader to [13] for a detailed formal definition of electronic voting protocols in applied pi calculus. Herein, we state the salient points of the definition. We assume that there is a sort $\mathsf{voteoption}$ in our signature which contains at least two constants (0-ary function symbols), denoted by $\mathbf{0}$ and $\mathbf{1}$, that do not occur in E. Furthermore, we assume that the protocol can be expressed as a parametric *plain* process $V(x_a, x_b)$ with two free variables x_a and x_b of the sort $\mathsf{voteoption}$.[4] For $\mathrm{v}_a, \mathrm{v}_b \in \{\mathbf{0}, \mathbf{1}\}$, the voter process $V(\mathrm{v}_a, \mathrm{v}_b)$ represents the process in which Alice and Bob vote for options v_a and v_b respectively. Although these assumptions are sufficient to model privacy as observational equivalence, the definition in terms of epistemic logic requires us to introduce events to annotate the individual voter preferences and consider all possible traces within a single process.

Towards this end we introduce a parametric event $\mathrm{votes}(\_,\_)$ with two arguments of the sort $\mathsf{voteoption}$ which is not present in the voting process $V(x_a, x_b)$. From now on, we consider the following voting process which considers all voting scenarios:

$$\mathcal{V} = \textstyle\sum_{v_a, v_b \in \{\mathbf{0},\mathbf{1}\}} \mathrm{votes}(\mathrm{v}_a, \mathrm{v}_b).V(\mathrm{v}_a, \mathrm{v}_b).$$

The process $\mathcal{V}$ shall henceforth be called a *voting process*.

Privacy as observational equivalence. We are ready to state the formalization of privacy as proposed in [13], which we shall call *strong privacy* for the rest of this section. Intuitively, the voting protocol represented as V respects strong privacy if the intruder cannot distinguish the two protocol instances in which Alice and Bob's votes are swapped.

Definition 8. *The voting process $\mathcal{V}$ respects* strong privacy *if* $V(\mathbf{0}, \mathbf{1}) \approx V(\mathbf{1}, \mathbf{0})$.

Privacy as epistemic formula. We need a few definitions to state privacy as an epistemic formula. An inspection of the construction of $\mathcal{V}$ shows that since the events votes do not occur in V, any maximal trace of $\mathcal{V}$ consists of only one event $\mathrm{votes}(\mathrm{v}_a, \mathrm{v}_b)$ in the store and corresponds to Alice and Bob voting for option v_a and v_b respectively. Also (from construction of the epistemic logic in Section 3), we assume that there is a binary predicate in our logic corresponding to the event votes which we shall (again in the interest of keeping the syntax simple) denote by votes. We also assume that there are two 0-ary function symbols corresponding to the two voting options which shall again denote by $\mathbf{0}$ and $\mathbf{1}$. Now, given $v \in \{\mathbf{0}, \mathbf{1}\}$ consider the formula

$$\mathsf{Avote}(v) = \mathrm{votes}(v, \mathbf{0}) \vee \mathrm{votes}(v, \mathbf{1}).$$

Intuitively the formula is true in a state reachable from $\mathcal{V}$ if Alice votes for option v. Similarly we can define formula $\mathsf{Bvote}(v)$.

Now, according to [4], a protocol respects *privacy for Alice* if the intruder cannot (epistemically) know which voting option Alice exercised. A protocol respects privacy if it respects privacy for both Alice and Bob. Please note that this definition does not usually hold for voting protocols in which the final tally of the votes are announced– a unanimous election always reveals each individual's vote. Hence, a more appropriate formulation is that whenever Alice and Bob vote differently, the intruder cannot learn how each of them voted. This gives us the following definition which states that intruder can learn how a voter voted only if the other voter voted the same option.

[4] V being a plain process is a simplification and we could have started with a non-empty frame.

Definition 9 (privacy). *The voting process $\mathcal{V}$ respects* privacy *if* $\mathcal{V} \models \mathsf{Aprivacy} \wedge \mathsf{Bprivacy}$ *where*

– $\mathsf{Aprivacy} \stackrel{\mathsf{def}}{=} \wedge_{\mathrm{v}\in\{\mathbf{0},\mathbf{1}\}}\Box(\mathsf{K}(\mathsf{Avote}(\mathrm{v})) \rightarrow \mathsf{Bvote}(\mathrm{v}))$*, and*
– $\mathsf{Bprivacy} \stackrel{\mathsf{def}}{=} \wedge_{\mathrm{v}\in\{\mathbf{0},\mathbf{1}\}}\Box(\mathsf{K}(\mathsf{Bvote}(\mathrm{v})) \rightarrow \mathsf{Avote}(\mathrm{v}))$.

Strong privacy implies privacy. We now show that privacy in terms of observational equivalence implies privacy in terms of epistemic formulas. In fact we show a stronger statement, namely, that if $V(\mathbf{0},\mathbf{1}) \approx_t V(\mathbf{1},\mathbf{0})$ then the protocol will respect privacy. The proof of the statement is deferred to the Appendix.

Theorem 1. *If $V(\mathbf{0},\mathbf{1}) \approx_t V(\mathbf{1},\mathbf{0})$ then the voting process $\mathcal{V}$ respects privacy. Hence, if $\mathcal{V}$ respects strong privacy then it respects privacy.*

Now, privacy in terms of epistemic formulas does not imply strong privacy. One can construct examples which respect privacy but not strong privacy, based on the fact that bisimulation is a finer relation than trace equivalence. However, a partial converse of Theorem 1 holds– under reasonable assumptions privacy implies $V(\mathbf{0},\mathbf{1}) \approx_t V(\mathbf{1},\mathbf{0})$.

Privacy implies trace equivalence. In order to state these assumptions, we need a few definitions. First we need the definition of a publishing trace. Intuitively, we say that a maximal trace tr is a publishing trace if the intruder learns which votes were cast (but not the link between the voters and individual votes) and can distinguish it from any other trace when the set of votes cast are different. For example, a publishing trace in which Alice and Bob vote $\mathbf{0}$ and $\mathbf{1}$ is distinguishable from one in which they cast $\mathbf{0}$ and $\mathbf{0}$ but not necessarily from one in which they cast $\mathbf{1}$ and $\mathbf{0}$ respectively. A maximal trace that is not publishing is said to be an abort trace. Intuitively, this says that the protocol could not be completed and hence votes are not published.[5]

Definition 10 (publishing and abort traces). *Given $\mathrm{v}_a, \mathrm{v}_b \in \{\mathbf{0},\mathbf{1}\}$, a maximal trace $\mathsf{tr} \in \mathsf{tr}_{\mathsf{max}}(V(\mathrm{v}_a, \mathrm{v}_b))$ is said to be a* publishing trace *if for any $\mathrm{v}'_a, \mathrm{v}'_b \in \{\mathbf{0},\mathbf{1}\}$ such that $\{\mathrm{v}_a, \mathrm{v}_b\} \neq \{\mathrm{v}'_a, \mathrm{v}'_b\}$, there is no $\mathsf{tr}' \in \mathsf{tr}(V(\mathrm{v}'_a, \mathrm{v}'_b))$ such that $\mathsf{tr} \sim_t \mathsf{tr}'$. Otherwise tr is an* abort trace.

We say that a protocol is equivalent for aborts if an abort trace can be mimicked irrespective of how Alice and Bob decided to vote.

Definition 11 (equivalent for aborts). *Given $\mathrm{v}_a, \mathrm{v}_b \in \{\mathbf{0},\mathbf{1}\}$ and $\mathsf{tr} \in \mathsf{tr}_{\mathsf{max}}(V(\mathrm{v}_a, \mathrm{v}_b))$ an abort trace. We say that $\mathcal{V}$ is* equivalent for aborts *if for any $\mathrm{v}'_a, \mathrm{v}'_b \in \{\mathbf{0},\mathbf{1}\}$ there is a $\mathsf{tr}' \in \mathsf{tr}_{\mathsf{max}}(V(\mathrm{v}'_a, \mathrm{v}'_b))$ such that $\mathsf{tr} \sim_t \mathsf{tr}'$.*

We have the partial converse of Theorem 1. The proof is deferred to the Appendix.

Theorem 2. *Let $\mathcal{V} = \sum_{v_a,v_b\in\{\mathbf{0},\mathbf{1}\}} \mathrm{votes}(\mathrm{v}_a, \mathrm{v}_b).V(\mathrm{v}_a, \mathrm{v}_b)$ be a voting process such that $\mathcal{V}$ is equivalent for aborts and respects privacy. Then $V(\mathbf{0},\mathbf{1}) \approx_t V(\mathbf{1},\mathbf{0})$.*

Theorem 1 and Theorem 2 suggest that trace-equivalence is the more appropriate notion for defining privacy of votes in electronic voting even though the bisimulation-based definition (which implies privacy) has better proof techniques.

[5] We believe that a good electronic voting protocol should not have abort traces. However, this property has not been studied in literature.

5 Related and Future Work

Related work. Several authors (e.g. [16, 12, 19]) have recognized the complementary nature of the process algebraic and epistemic approaches and the benefit to combine them. Different approaches have been proposed to bridge this gap. In [16], *function views* are used to represent partial information and make the interface between protocol and properties. In order to get epistemic specifications closer to a behavioral specification, van Eijck and Orzan [19] propose a dynamic epistemic logic. However, it seems that no mediation is necessary [15, 12] and it is possible to bridge this gap by proposing a combined framework as it is also suggested in this paper. However, in the works cited above, the authors study abstract versions of protocols which do not take into account cryptographic primitives (e.g. encryption, signature, ...) and their specific properties.

Some recent works [17, 10] have been devoted to designing a logic to characterize *static equivalence*. In [17], they build upon the logic for frames and extend it with Hennessy-Milner modalities, yielding a logic for applied pi processes which characterizes labeled bisimilarity. However, as we already pointed out in the Introduction, our goal is different and we want to define a logic that is expressive enough to state a variety of security properties in a natural way. The advantage of this approach is evident in our example of formalizing privacy in e-voting protocols in which we were able to establish the exact relationship between two formal definitions of privacy in e-voting protocols.

Another similarity between our work and the work in [10] is that they also have epistemic modalities. The work in [10] has another advantage in that they reason about multiple agents and hence their logic has epistemic modalities for multiple agents and not just the intruder. This is however achieved by interpreting the logic over an agent-indexed family of frames with a frame representing the set of messages in an agent's possessions. Since they are mostly interested in studying static equivalence, they do not mention how these frames are obtained. An applied pi-calculus process only keeps track of the messages in intruder's possession and thus we have only one epistemic modality.

The problem of having a suitable language which allows for an expressive property logic is a well-known problem in the context of cryptographic protocols verification. In [7, 11], such a language and logic is proposed and allows specification of a large class of security properties. However, none of the underlying protocol languages is as expressive as the applied pi calculus. We are able to model a large class of protocols which may use less classical cryptographic primitives, specified by an equational theory, in an intuitive way. Therefore, our framework can be used for protocols such as electronic voting protocols, contract signing protocols, ...

Future Work. The formalism presented in this paper is a starting point, and we intend to study stronger anonymity properties such as coercion-resistance that arise in security protocols. Another line of investigation is to extend the formalism to allow for reasoning about epistemic knowledge of multiple agents, and this would involve extension of both the calculus and the logic. We also intend to study model-checking algorithms to verify whether a process satisfies a given formula. Finally, we also intend to investigate an axiomatization of the logic presented in the paper.

References

1. M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. *Theoretical Computer Science*, 387(1-2):2–32, 2006.
2. M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proc. 28th Symposium on Principles of Programming Languages*, pages 104–115, 2001.
3. M. Backes, M. Maffei, and D. Unruh. Zero-knowledge in the applied pi-calculus and automated verification of the direct anonymous attestation protocol. In *Proc. 29th IEEE Symposium on Security and Privacy*, 2008.
4. A. Baskar, R. Ramanujam, and S. P. Suresh. Knowledge-based modelling of voting protocols. In *Proc. 11th Conference on Theoretical Aspects of Rationality and Knowledge*, pages 62–71, 2007.
5. B. Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *Proc. 14th Computer Security Foundations Workshop*, pages 82–96, 2001.
6. B. Blanchet. From Secrecy to Authenticity in Security Protocols. In *9th International Static Analysis Symposium*, pages 342–359, 2002.
7. J. Borgström, S. Kramer, and U. Nestmann. Calculus of Cryptographic Communication. In *Proc. Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis*, 2006.
8. M. Burrows, M. Abadi, and R. M. Needham. A logic of authentication. *ACM Trans. Comput. Syst.*, 8(1):18–36, 1990.
9. R. Chadha, S. Kremer, and A. Scedrov. Formal analysis of multi-party contract signing. *Journal of Automated Reasoning*, 36(1-2):39–83, 2006.
10. M. Cohen and M. Dam. A complete axiomatization of knowledge and cryptography. In *Proc. 22nd IEEE Symposium on Logic in Computer Science*, pages 77–88, 2007.
11. R. Corin, A. Saptawijaya, and S. Etalle. PS-LTL for constraint-based security protocol analysis. In *Proc. 21st International Conference on Logic Programming*, pages 439–440, 2005.
12. F. Dechesne, M. R. Mousavi, and S. Orzan. Operational and epistemic approaches to protocol analysis: Bridging the gap. In *Proc. 14th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, pages 226–241, 2007.
13. S. Delaune, S. Kremer, and M. D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 2008. To appear.
14. R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995.
15. J. Y. Halpern and K. R. O'Neill. Anonymity and information hiding in multiagent systems. *Journal of Computer Security*, 13(3):483–512, 2005.
16. D. Hughes and V. Shmatikov. Information hiding, anonymity and privacy: a modular approach. *Journal of Computer Security*, 12(1):3–36, 2004.
17. H. Hüttel and M. D. Pedersen. A logical characterisation of static equivalence. *Electr. Notes Theor. Comput. Sci.*, 173:139–157, 2007.
18. H. Jonker and W. Pieters. Receipt-freeness as a special case of anonymity in epistemic logic. In *Proc. IAVoSS Workshop On Trustworthy Elections*, 2006.
19. J. van Eijck and S. Orzan. Epistemic verification of anonymity. *Electr. Notes Theor. Comput. Sci.*, 168:159–174, 2007.
20. T. Y. C. Woo and S. S. Lam. A semantic model for authentication protocols. In *Proc. 14th IEEE Symposium on Security and Privacy*, 1993.

A Definition of α-renaming of traces

Definition 12 **(α equivalence of traces ($\sim_\alpha$)).** *Given a trace* $\mathsf{tr} = A_0 \xrightarrow{\gamma_1} A_1 \ldots \xrightarrow{\gamma_\ell} A_n$ *we say that a name n is bound at position j if $n \in bn(\gamma_j)$. Pick $n' \notin \cup_{\ell=0}^{n}(bn(\gamma_k) \cup fn(\gamma_k) \cup bn(A_k) \cup fn(A_k))$ and let $j' > j$ be the smallest number such that n is bound at position j' (if no such number occurs let $j' = \ell + 1$).*

Let tr' be the trace obtained from tr by replacing all occurrences of n by n' in γ_k for $j \leq k < j'$ and by replacing A_k by $A_k\{n'/n\}$ for $j < k \leq j'$. We say that tr' is obtained from tr by an α-conversion of n by n'. We write $\mathsf{tr} \to_\alpha \mathsf{tr}'$ if there exist some n and n' such that tr' is obtained from tr by an α-conversion of n by n'.

The relation $\sim_\alpha$ is defined to be the reflexive, symmetric and transitive closure of $\to_\alpha$.

The reason for defining α-equivalence is illustrated in the following example.

Example 10. Consider the processes

$$A = \nu c.\text{out}(a, c).\text{in}(c, x).0$$

and

$$B = \nu d.\nu n.\text{out}(a, d).\text{in}(d, x).\text{if } (x = n) \text{ then out}(a, c).$$

The two processes A and B are observationally equivalent. Assume further that the vocabulary consists of a unary function symbol hash and a constant 0 of the base type. The process A exhibits the trace tr in $A \xrightarrow{\nu c.\text{out}(a,c)} \text{in}(c, x).0 \xrightarrow{\text{in}(c,\mathsf{hash}(0))} 0$. Note that although B is observationally equivalent to A, B cannot exhibit the same trace tr as c occurs free in B. However, note that the trace tr is α-equivalent to the trace $\mathsf{tr}' = A \xrightarrow{\nu e.\text{out}(a,e)} \text{in}(e, x).0 \xrightarrow{\text{in}(e,\mathsf{hash}(0))} 0$ and B can exhibit tr'.

B Proof of Theorem 1

Theorem 1. *If $V(\mathbf{0}, \mathbf{1}) \approx_t V(\mathbf{1}, \mathbf{0})$ then the voting process $\mathcal{V}$ respects privacy. Hence, if $\mathcal{V}$ respects strong privacy then it respects privacy.*

Proof. The epistemic formula that expresses privacy is made of four conjuncts. We shall show that if $V(\mathbf{0}, \mathbf{1}) \approx_t V(\mathbf{1}, \mathbf{0})$ then $\mathcal{V}$ satisfies one conjunct, say $\Box(\mathsf{K}(\mathsf{Avote}(\mathbf{0})) \to \mathsf{Bvote}(\mathbf{0}))$. The proof that $\mathcal{V}$ satisfies other conjuncts is similar.

Now consider any maximal trace $\mathsf{tr} \in \mathsf{tr}_{\mathsf{max}}(\mathcal{V})$ such that there is a $j \leq |\mathsf{tr}|$ such that $\mathcal{V}, \mathsf{tr}, j \models \mathsf{K}(\mathsf{Avote}(\mathbf{0}))$. It suffices to show that $\mathcal{V}, \mathsf{tr}, j \models \mathsf{Bvote}(\mathbf{0})$. Since $\mathcal{V}, \mathsf{tr}, j \models \mathsf{K}(\mathsf{Avote}(\mathbf{0}))$, it must be the case $\mathcal{V}, \mathsf{tr}, j \models \mathsf{Avote}(\mathbf{0})$, that is, $\mathcal{V}, \mathsf{tr}, j \models \text{votes}(\mathbf{0}, \mathbf{0}) \vee \text{votes}(\mathbf{0}, \mathbf{1})$. We need to show that $\mathcal{V}, \mathsf{tr}, j \models \mathsf{Bvote}(\mathbf{0})$. Hence it suffices to show that $\mathcal{V}, \mathsf{tr}, j \not\models \text{votes}(\mathbf{0}, \mathbf{1})$.

We proceed by contradiction. Assume that $\mathcal{V}, \mathsf{tr}, j \models \text{votes}(\mathbf{0}, \mathbf{1})$. From the fact that the event votes does not occur in V and the construction of $\mathcal{V}$, we can show that $j \geq 2$ and the trace tr consists of i initial internal steps of $\mathcal{V}$ ($2 \leq i \leq 4$, i.e., $i - 1$

choices and 1 event) leading to $([\mathrm{votes}(\mathbf{0},\mathbf{1})] \mid V(\mathbf{0},\mathbf{1}))$ followed by a maximal trace of $([\mathrm{votes}(\mathbf{0},\mathbf{1})] \mid V(\mathbf{0},\mathbf{1}))$.

Now one can show easily that $V(\mathbf{0},\mathbf{1}) \approx_t V(\mathbf{1},\mathbf{0})$ iff $([\mathrm{votes}(\mathbf{0},\mathbf{1})] \mid V(\mathbf{0},\mathbf{1})) \approx_t ([\mathrm{votes}(\mathbf{1},\mathbf{0})] \mid V(\mathbf{1},\mathbf{0}))$. Thus there is a trace $\mathsf{tr}_1 \in \mathsf{tr}([\mathrm{votes}(\mathbf{1},\mathbf{0})] \mid V(\mathbf{1},\mathbf{0}))$ such that $\mathsf{tr}[i, |\mathsf{tr}|] \sim_t \mathsf{tr}_1$. Consider the trace tr' of $\mathcal{V}$ which starts with two internal actions (choice and event) leading to $([\mathrm{votes}(\mathbf{1},\mathbf{0})] \mid V(\mathbf{1},\mathbf{0}))$ followed by tr_1. Now $\mathsf{tr} \sim_t \mathsf{tr}'$. By definition of equivalence of traces, we get that there must be a $k \leq |\mathsf{tr}'|$ such that $\mathsf{tr}[0,j] \sim_t \mathsf{tr}'[0,k]$.

Now, note that since $j \geq 2$ and first two actions of tr and tr' are internal actions, we can show that $k \geq 2$. Using the fact that tr' consists of two internal steps followed by a trace of $([\mathrm{votes}(\mathbf{1},\mathbf{0})] \mid V(\mathbf{1},\mathbf{0}))$ and the fact that the event $\mathrm{votes}(\mathbf{0},\mathbf{1})$ does not occur in V, we get that $\mathsf{tr}'[k] \not\models \mathsf{Avote}(\mathbf{0})$. Hence, for any maximal tr_0 that extends tr' we get $\mathcal{V}, \mathsf{tr}_0, k \not\models \mathsf{Avote}(\mathbf{0})$. On the other hand, since $\mathsf{tr}[0,j] \sim_t \mathsf{tr}_0[0,k]$ and $\mathcal{V}, \mathsf{tr}, j \models \mathsf{K}(\mathsf{Avote}(\mathbf{0}))$, we have $\mathcal{V}, \mathsf{tr}_0, k \models \mathsf{Avote}(\mathbf{0})$. A contradiction. □

C Proof of Theorem 2

Theorem 2. *Let* $\mathcal{V} = \sum_{v_a, v_b \in \{\mathbf{0},\mathbf{1}\}} \mathrm{votes}(\mathrm{v}_a, \mathrm{v}_b).V(\mathrm{v}_a, \mathrm{v}_b)$ *be a voting process such that* $\mathcal{V}$ *is equivalent for aborts and respects privacy. Then* $V(\mathbf{0},\mathbf{1}) \approx_t V(\mathbf{1},\mathbf{0})$.

Proof. Given a trace $\mathsf{tr} \in \mathsf{tr}_{\mathsf{max}}(V(\mathrm{v}_a, \mathrm{v}_b))$ we define $[\mathrm{votes}(\mathrm{v}_a, \mathrm{v}_b)] \mid \mathsf{tr}$ to be the trace obtained by replacing $\mathsf{tr}[j]$ by $[\mathrm{votes}(\mathrm{v}_a, \mathrm{v}_b)] \mid \mathsf{tr}[j]$ for each $0 \leq j \leq |\mathsf{tr}.|$ We say that $[\mathrm{votes}(\mathrm{v}_a, \mathrm{v}_b)] \mid \mathsf{tr}$ is publishing (abort) trace if tr is a publishing (abort) trace. First we observe that it can be shown using the fact that events do not affect trace equivalence, that $V(\mathbf{0},\mathbf{1}) \approx_t V(\mathbf{1},\mathbf{0})$ iff $[\mathrm{votes}(\mathbf{0},\mathbf{1})] \mid V(\mathbf{0},\mathbf{1}) \approx_t [\mathrm{votes}(\mathbf{1},\mathbf{0})] \mid V(\mathbf{1},\mathbf{0})$. We now proceed by contradiction. Assume that $[\mathrm{votes}(\mathbf{0},\mathbf{1})] \mid V(\mathbf{0},\mathbf{1}) \not\approx_t [\mathrm{votes}(\mathbf{1},\mathbf{0})] \mid V(\mathbf{1},\mathbf{0})$. Thus either $[\mathrm{votes}(\mathbf{0},\mathbf{1})] \mid V(\mathbf{0},\mathbf{1}) \not\sqsubseteq_t [\mathrm{votes}(\mathbf{1},\mathbf{0})] \mid V(\mathbf{1},\mathbf{0})$ or $[\mathrm{votes}(\mathbf{1},\mathbf{0})] \mid V(\mathbf{1},\mathbf{0}) \not\sqsubseteq_t [\mathrm{votes}(\mathbf{0},\mathbf{1})] \mid V(\mathbf{0},\mathbf{1})$.

Assume that $[\mathrm{votes}(\mathbf{0},\mathbf{1})] \mid V(\mathbf{0},\mathbf{1}) \not\sqsubseteq_t [\mathrm{votes}(\mathbf{1},\mathbf{0})] \mid V(\mathbf{1},\mathbf{0})$ (the other case is similar). Thus there is a trace $\mathsf{tr} \in \mathsf{tr}([\mathrm{votes}(\mathbf{0},\mathbf{1})] \mid V(\mathbf{0},\mathbf{1}))$ such that for all $\mathsf{tr}' \in \mathsf{tr}([\mathrm{votes}(\mathbf{1},\mathbf{0})] \mid V(\mathbf{1},\mathbf{0}))$, we have that $\mathsf{tr} \not\sim_t \mathsf{tr}'$. Furthermore, we can assume tr to be maximal (otherwise we can extend tr to a maximal trace). Now since the $\mathcal{V}$ is equivalent for aborts, tr cannot be an abort trace.

Now consider the maximal trace tr_0 of $\mathcal{V}$ which consists of two internal actions followed by the trace tr. We have that $\mathsf{tr}_0(|\mathsf{tr}_0|) \models \mathrm{votes}(\mathbf{0},\mathbf{1})$. Now, pick any other maximal trace tr_1 of $\mathcal{V}$ such that there is a $j \leq |\mathsf{tr}_1|$ such that $\mathsf{tr}_1[0,j] \sim_t \mathsf{tr}_0$. Any maximal trace of $\mathcal{V}$ consists of two internal steps followed by a maximal trace of $[\mathrm{votes}(\mathrm{v}_a, \mathrm{v}_b)] \mid V(\mathrm{v}_a, \mathrm{v}_b)$ for some $\mathrm{v}_a, \mathrm{v}_b$. Since the first two steps of tr_0 and tr_1 are internal and $j > 2$, we can show that $\mathsf{tr}_1[2,j] \sim_t \mathsf{tr}_0[2, |\mathsf{tr}_0|]$.

Now $\mathsf{tr}_0[2, |\mathsf{tr}_0|]$ is tr. Furthermore $\mathsf{tr}_1[2,j] \in \mathsf{tr}([\mathrm{votes}(\mathrm{v}_a, \mathrm{v}_b)] \mid V(\mathrm{v}_a, \mathrm{v}_b))$ for some $\mathrm{v}_a, \mathrm{v}_b$. Since tr is a publishing trace and tr is not contained (upto equivalence) in $\mathsf{tr}([\mathrm{votes}(\mathbf{1},\mathbf{0})] \mid V(\mathbf{1},\mathbf{0}))$, we can conclude that $\mathsf{tr}_1[2,j]$ is a trace of $[\mathrm{votes}(\mathbf{0},\mathbf{1})] \mid V(\mathbf{0},\mathbf{1})$. Hence $\mathsf{tr}_1[j] \models \mathrm{votes}(\mathbf{0},\mathbf{1})$. Hence, we have that $\mathcal{V}, \mathsf{tr}_0, |\mathsf{tr}_0| \models \mathsf{K}(\mathsf{Avote}(\mathbf{0}))$ but $\mathcal{V}, \mathsf{tr}_0, |\mathsf{tr}_0| \not\models \mathsf{Bvote}(\mathbf{0})$. A contradiction. □