Complexity of Anonymity for Security Protocols

Ferucio Laurenţiu Ţiplea, Loredana Vamanu, and Cosmin Vârlan

Department of Computer Science
"Al.I.Cuza" University of Iaşi
Iaşi 700506, Romania
{fltiplea,loredana.vamanu,vcosmin}@info.uaic.ro

Abstract. Anonymity, as an instance of information hiding, is one of the security properties intensively studied nowadays due to its applications to various fields such as e-voting, e-commerce, e-mail, e -cash, and so on. In this paper we study the decidability and complexity status of the anonymity property in security protocols. We show that anonymity is undecidable for unrestricted security protocols, is NEXPTIME-complete for bounded security protocols, and it is NP-complete for 1-session bounded security protocols. In order to reach these objectives, an epistemic language and logic to reason about anonymity properties for security protocols under an active intruder, are provided. Agent states are endowed with facts derived from actions performed by agents in protocol executions, and an inference system is provided. To define anonymity, an observational equivalence is used, which is shown to be decidable in deterministic polynomial time.

1 Introduction

Anonymity, as an instance of *information hiding*, is one of the security properties intensively studied nowadays due to its applications to various fields such as electronic voting, electronic commerce, electronic mail, electronic cash and so on. It embraces many forms, such as *sender* or *receiver anonymity*, and it is closely related to *unlinkability*, *indistinguishability*, and *role interchangeability* [12, 10, 17]. The intuition behind anonymity is that an agent who performed some action is not "identifiable" by some observer of the system. "Non-identifiability" might mean that the observer is not able to see that the agent performed that action, or he saw that many other agents performed that action.

Several approaches to model anonymity have been proposed, such as [15,16,8, 7,5,9]. The approach in [15] is CSP-based, while the ones in [16,7] are based on epistemic logics. The authors in [16] show, in an epistemic logic framework, how the agent states can be augmented with information about actions performed by agents during protocol computations, and then propose an inference mechanism by which more information can be deduced. Several anonymity concepts are then proposed and discussed. The epistemic approach in [7] models anonymity in a multi-agent system framework. This is a very nice and general approach to talk about anonymity-related properties and many other papers on anonymity built

D. Gritzalis, B. Preneel, and M. Theoharidou (Eds.): ESORICS 2010, LNCS 6345, pp. 558–572, 2010.

on it [10,17]. Based on the concept of a *function view* as a concise representation of the intruder's partial knowledge about a function, Hughes and Shmatikov have proposed a rich variety of anonymity-related properties in [8]. The *cryptographic protocol logic* (CPL) in [9] came as an ambitious general framework for formalizing a very large class of security properties, including anonymity as well. While CPL seems very expressive, the model checking problem for it is undecidable and not too much about decidable fragments and proof systems for the core CPL is known.

From a computational point of view, the anonymity problem for security protocols is a decision problem: it is the problem to decide, given a security protocol and an action of it, whether or not the action is anonymous with respect to some agent. None of the papers mentioned above discusses the decidability and complexity status of this problem. As anonymity is not a *trace-based property* but it is based on an *observational equivalence* on protocol states, it is expected that anonymity is harder than secrecy or authentication. This is because, given a state of the protocol which is to be checked against some property, it might be the case that all observationally equivalent states are needed to be analyzed in order to decide the property.

In this paper we study the decidability and complexity status of the anonymity property for security protocols. Thus, we show that anonymity is undecidable for unrestricted security protocols, is NEXPTIME-complete for bounded security protocols, and it is NP-complete for 1-session bounded security protocols. In order to reach these objectives we enrich the security protocol model in [13, 19] by adding *facts* to agent states. Then we develop an inference system by which agents can infer more properties from facts. This inference system has special constructs, mainly due to the fact that in our approach the intruder is active, and this makes it different from the one in [16] (if the intruder is passive, then any receiver knows exactly from whom the message he received comes). To define anonymity, an *observational equivalence* is used, which is decidable in deterministic polynomial time.

The paper is organized in five sections. The formal model we use in this paper for security protocols is introduced in Section 2. Facts, as a way to cope with information about actions performed by agents in a security protocol, are introduced in Section 3, together with an inference system. Our observational equivalence is also a topic of this section, as well as the anonymity concepts we use in the paper. It is shown that the observational equivalence is decidable in deterministic polynomial time. Section 4 presents the main results of the paper. We conclude in Section 5.

2 Modeling Security Protocols

We recall the formalism in [13] with slight modifications [19], and use it in order to develop the main results of the paper.

Protocol signatures and terms. A *security protocol signature* is a 3-tuple $\mathcal{S} = (\mathcal{A}, \mathcal{K}, \mathcal{N})$ consisting of a finite set $\mathcal{A}$ of *agent names* (or shortly, *agents*)

and two at most countable sets $\mathcal{K}$ and $\mathcal{N}$ of *keys* and *nonces*, respectively. It is assumed that:

- $\mathcal{A}$ contains a special element denoted by I and called the *intruder*. All the other elements are called *honest agents* and Ho denotes their set;
- $\mathcal{K} = \mathcal{K}_0 \cup \mathcal{K}_1$, where $\mathcal{K}_0$ is the set of *short-term keys* and $\mathcal{K}_1$ is a finite set of *long-term keys*. The elements of $\mathcal{K}_1$ are of the form K_A^e (A's public key), or K_A^d (A's private key), or K_{AB} (shared key by A and B), where A and B are distinct agents;
- some honest agents A may be provided from the beginning with some *secret information* $Secret_A \subseteq \mathcal{K}_0 \cup \mathcal{N}$, not known to the intruder. $Secret_A$ does not contain long-term keys because they will never be communicated by agents during the runs;
- the intruder is provided from the beginning with a set of nonces $\mathcal{N}_I \subseteq \mathcal{N}$ and a set of short-term keys $\mathcal{K}_{0,I} \subseteq \mathcal{K}_0$. It is assumed that no elements in $\mathcal{N}_I \cup \mathcal{K}_{0,I}$ can be generated by honest agents.

The set of *basic terms* is $\mathcal{T}_0 = \mathcal{A} \cup \mathcal{K} \cup \mathcal{N}$. The set $\mathcal{T}$ of *terms* is defined inductively: every basic term is a term; if t_1 and t_2 are terms, then (t_1, t_2) is a term; if t is a term and K is a key, then $\{t\}_K$ is a term. We extend the construct (t_1, t_2) to $(t_1, \ldots, t_n)$ as usual by letting $(t_1, \ldots, t_n) = ((t_1, \ldots, t_{n-1}), t_n)$, for all $n \geq 3$. Sometimes, parenthesis will be omitted. Given a term t, $Sub(t)$ is the set of all *subterms* of t (defined as usual). This notation is extended to sets of terms by union.

The length of a term is defined as usual, by taking into consideration that pairing and encryption are operations. Thus, $|t| = 1$ for any $t \in \mathcal{T}_0$, $|(t_1, t_2)| = |t_1| + |t_2| + 1$, for any terms t_1 and t_2, and $|\{t\}_K| = |t| + 2$, for any term t and key K.

The *perfect encryption assumption* we adopt [1] states that a message encrypted with a key K can be decrypted only by an agent who knows the corresponding inverse of K (denoted K^{-1}), and the only way to compute $\{t\}_K$ is by encrypting t with K.

Actions. There are two types of actions, send and receive. A *send action* is of the form $A!B : (M)t$, and a *receive action* is of the form $A?B : t$. In both cases, A is assumed an honest agent who *performs the action*, $A \neq B$, $t \in \mathcal{T}$ is the *term of the action*, and $M \subseteq Sub(t) \cap (\mathcal{N} \cup \mathcal{K}_0)$ is the *set of new terms of the action*.

$M(a)$ denotes M, if $a = A!B : (M)t$, and the empty set, if $a = A?B : t$; $t(a)$ stands for the term of a. When $M = \emptyset$ we will simply write $A!B : t$. For a sequence of actions $w = a_1 \cdots a_l$ and an agent A, define the *restriction of w to A*, denoted $w|_A$, as being the sequence obtained from w by removing all actions not performed by A. The notations $M(a)$ and $t(a)$ are extended to sequences of actions by union.

Protocols. A *security protocol* (or simply, *protocol*) is a triple $\mathcal{P} = (\mathcal{S}, \mathcal{C}, w)$, where $\mathcal{S}$ is a security protocol signature, $\mathcal{C}$ is a subset of $\mathcal{T}_0$, called the set of *constants* of $\mathcal{P}$, and w is a non-empty sequence of actions, called the *body* of the

protocol, such that no action in w contains the intruder. Constants are publicly known elements in the protocol that cannot be re-instantiated (as it will be explained below). As usual, $\mathcal{C}$ does not include private keys, elements in $Secret_A$ for any honest agent A, or elements in $\mathcal{N}_I$, $\mathcal{K}_{0,I}$ and $M(w)$.

Any non-empty sequence $w|_A$, where A is an agent, is called a *role* of the protocol. A role specifies the actions a participant should perform in a protocol, and the order of these actions.

Substitutions and events. Instantiations of a protocol are given by *substitutions*, which are functions σ that map agents to agents, nonces to arbitrary terms, short-term keys to short-term keys, and long-term keys to long-term keys. Moreover, for long-term keys, σ should satisfy $\sigma(K_A^e) = K_{\sigma(A)}^e$, $\sigma(K_A^d) = K_{\sigma(A)}^d$, and $\sigma(K_{AB}) = K_{\sigma(A)\sigma(B)}$, for any distinct agents A and B.

Substitutions are homomorphically extended to terms, actions, and sequences of actions. A substitution σ is called *suitable for an action* $a = AxB : y$ if $\sigma(A)$ is an honest agent, $\sigma(A) \neq \sigma(B)$, and σ maps distinct nonces from $M(a)$ into distinct nonces, distinct keys into distinct keys, and it has disjoint ranges for $M(a)$ and $Sub(t(a)) - M(a)$. σ is *suitable for a sequence of actions* if it is suitable for each action in the sequence, and σ is *suitable for a subset* $C \subseteq \mathcal{T}_0$ if it is the identity on C.

An *event* of a protocol $\mathcal{P} = (\mathcal{S}, \mathcal{C}, w)$ is any triple $e_i = (u, \sigma, i)$, where $u = a_1 \cdots a_l$ is a role of $\mathcal{P}$, σ is a substitution suitable for u and $\mathcal{C}$, and $1 \leq i \leq l$. $\sigma(a_i)$ is the *action of the event* e_i. As usual, $act(e_i)$ ($t(e_i)$, $M(e_i)$) stands for the the action of e_i (term of e_i, set of new terms of e_i). The *local precedence relation* on events is defined by $(u, \sigma, i) \rightarrow (u', \sigma', i')$ if and only if $u' = u$, $\sigma' = \sigma$, and $i' = i+1$, provided that $i < |u|$. $\stackrel{+}{\rightarrow}$ is the transitive closure of $\rightarrow$. Given an event e, ${}^\bullet e$ stands for the *set of all local predecessors of* e, i.e., ${}^\bullet e = \{e' | e' \stackrel{+}{\rightarrow} e\}$.

Message generation rules. Given X a set of terms, $analz(X)$ stands for the least set which includes X, contains t_1 and t_2 whenever it contains (t_1, t_2), and contains t whenever it contains $\{\{t\}_K\}_{K^{-1}}$ or $\{t\}_K$ and K^{-1}. By $synth(X)$ we denote the least set which includes X, contains (t_1, t_2), for any terms $t_1, t_2 \in synth(X)$, and contains $\{t\}_K$, for any term t and key K in $synth(X)$. Moreover, $\overline{X}$ stands for $synth(analz(X))$.

States and runs. A *state* of a protocol $\mathcal{P}$ is an indexed set $s = (s_A | A \in \mathcal{A})$, where $s_A \subseteq \mathcal{T}$, for any agent A. The *initial state* is $s_0 = (s_{0A} | A \in \mathcal{A})$, where $s_{0A} = \mathcal{A} \cup \mathcal{C} \cup \mathcal{K}_A \cup Secret_A$ for any $A \in Ho$, $s_{0I} = \mathcal{A} \cup \mathcal{C} \cup \mathcal{K}_I \cup \mathcal{N}_I \cup \mathcal{K}_{0,I}$, and $\mathcal{K}_X$ is the set of long-term keys known by $X \in \mathcal{A}$.

For two states s and s' and an action a, we write $s[a\rangle s'$ if and only if:

1. if a is of the form $A!B : (M)t$, then:
   (a) $t \in \overline{s_A \cup M}$ and $M \cap Sub(s) = \emptyset$; (*enabling condition*)
   (b) $s'_A = s_A \cup M \cup \{t\}$, $s'_I = s_I \cup \{t\}$, and $s'_C = s_C$ for any $C \in \mathcal{A} - \{A, I\}$;
2. if a is of the form $A?B : t$, then:
   (a) $t \in \overline{s_I}$; (*enabling condition*)
   (b) $s'_A = s_A \cup \{t\}$ and $s'_C = s_C$, for all $C \neq A$.

We extend the notation "$[\cdot\rangle$" to events by letting $s[e\rangle s'$ whenever $s[act(e)\rangle s'$, and we call $s[e\rangle s'$ a *computation step*. A *computation* or *run* of a security protocol is any sequence $s_0[e_1\rangle s_1 \cdots s_{k-1}[e_k\rangle s_k$ of computation steps, also written as $s_0[e_1 \cdots e_k\rangle s$ or even $e_1 \cdots e_k$, such that $s_{i-1}[e_i\rangle s_i$ for any $1 \leq i \leq k$, and ${}^\bullet e_i \subseteq \{e_1, \ldots, e_{i-1}\}$ for any $1 \leq i \leq k$ (for $i = 1$, ${}^\bullet e_i$ should be empty).

3 Anonymity-Related Security Properties

In this section we show how the model presented in the previous section can be endowed with information necessary to define and reason about anonymity properties in security protocols. The main idea is to add *facts* to agent states once the agents perform actions in the protocol. Each agent may then deduce new facts by using his knowledge at some point in the protocol. Although the idea of endowing agent states by facts was already used in [16], our approach is different. We endow the agent states with less information but sufficient to define and reason about a large spectrum of anonymity properties. While [16] assumes a passive intruder, in our approach the intruder is active. This asks for special deduction rules, making the deduction process more complex.

To define anonymity, a state-based observational equivalence is used in our paper. Two states are observationally equivalent w.r.t. an agent if the agent can derive the same "meaningful information" from each of the states. The anonymity concepts in [16] are not based on any observational equivalence. Halpern and O'Neill's approach to anonymity [7] is a very general one, so the observational equivalence is not precisely defined in their paper. Precise observational equivalences have been proposed, but for particular classes of anonymous communication [3]. The observational equivalence in [5] is trace-based. However, anonymity is not a *trace-based property* (or, at least, it cannot be naturally defined as a trace-based property such as secrecy or authentication).

3.1 Augmenting Agent States with Facts

When an agent in a security protocol performs a send or a receive action, he may record a number of important pieces of information. These pieces of information can be formalized by using *facts*[1], that is, sentences of the form $P(t_1, \ldots, t_i)$, where P is a predicate symbol of arity at least one and $t_1, \ldots, t_i$ are message terms (facts beginning by the same predicate symbol P will also be called *P-facts*).

In order to exemplify this we shall consider a running example. In the protocol in Figure 1, the agent A asks B for a ticket to access some network service H guarded by some agent C. Once A gets the authenticated ticket from B, it sends it to C together with an encrypted copy for H. C checks the ticket and then sends the encrypted copy to H.

Four classes of information pieces are to be recorded by agents in our formalism:

[1] Later in this section, facts will be considered *primitive propositions* for defining the epistemic logic we use to talk about anonymity properties.

$$
\begin{aligned}
A\,!\,B &: (\{N_A, K\})\ \{A, B, H, N_A, K\}_{K_B^e} \\
B\,?\,A &: \{A, B, H, N_A, K\}_{K_B^e} \\
B\,!\,A &: \{N_A, B, Ticket\}_K, \{N_A, B, Ticket\}_{K_B^d} \\
A\,?\,B &: \{N_A, B, Ticket\}_K, \{N_A, B, Ticket\}_{K_B^d} \\
A\,!\,C &: \{Ticket, \{Ticket\}_{K_{AH}}\}_{K_{AC}} \\
C\,?\,A &: \{Ticket, \{Ticket\}_{K_{AH}}\}_{K_{AC}} \\
C\,!\,H &: \{\{Ticket\}_{K_{AH}}\}_{K_{CH}} \\
H\,?\,C &: \{\{Ticket\}_{K_{AH}}\}_{K_{CH}}
\end{aligned}
$$

Fig. 1. A running example

1. *sent*-facts. Each agent X who sends a message t to some agent Y records a fact $sent(X, t, Y)$. For instance, when the first action of our example will be performed, A records $sent(A, \{A, B, H, N_A, K\}_{K_B^e}, B)$;
2. *rec*-facts. According to the intruder type, two cases are to be considered:
   (a) *passive intruder*. If an action $X\,?\,Y : t$ was performed by X, then X may safely record a fact $rec(X, t, Y)$ because he knows that the message he received is from Y. For instance, if action two in our running example was performed in some computation, then B may record the fact $rec(B, \{A, B, H, N_A, K\}_{K_B^e}, A)$;
   (b) *active intruder*. If an action $X\,?\,Y : t$ was performed by X, then X might not be sure whether t comes from Y or from the intruder. In such a case X records a fact $rec(X, t, (Y, I))$ showing that t may be from Y or from I. For instance, if action two in our running example was performed in some computation, then B records the fact $rec(B, \{A, B, H, N_A, K\}_{K_B^e}, (A, I))$;
3. *gen*-facts. The message in the first action of our running example is *generated by A for B* because it is encrypted by B's public key; denoted this by $gen(A, \{A, B, H, N_A, K\}_{K_B^e}, B)$ and record it in A's state. Similarly, $\{Ticket\}_{K_{AH}}$ in the fifth action is *generated by A for H* because it is encrypted by a key shared by A and H. Therefore, $gen(A, \{Ticket\}_{K_{AH}}, H)$ will be recorded in A's state;
4. *auth*-facts. In the third action of the protocol, the message sent by B to A contains a sub-message of the form $\{N_A, B, Ticket\}_{K_B^d}$. This is in fact B's digital signature on the message $(N_A, B, Ticket)$; we denote this by $auth(B, (N_A, B, Ticket, \{N_A, B, Ticket\}_{K_B^d}))$ and record it in B's state.

We will now formalize our discussion above. First, we extend the concept of an agent state from Section 2 as follows. A state of an agent A is a pair of sets $s_A = (s_{A,m}, s_{A,f})$, where $s_{A,m}$ is a set of messages and $s_{A,f}$ is a set of facts. Intuitively, $s_{A,m}$ represents the set of all messages the agent A sent or received in some computation ξ from the initial state to the state s_A, and $s_{A,f}$ represents the set of facts which give information about the actions the agent A performed in ξ.

Then, a protocol state is of the form $s = (s_A | A \in \mathcal{A})$, where each s_A has the form $s_A = (s_{A,m}, s_{A,f})$. We naturally extend the notation Sub for terms and

sets of terms to agent states by $Sub(s_A) = Sub(s_{A,m})$, and to protocol states by $Sub(s) = \bigcup_{A \in \mathcal{A} - \{I\}} Sub(s_A)$.

The protocol computation rule has to be changed accordingly. Given two states s and s' and an action a, we write $s[a\rangle s'$ if and only if:

1. if a is of the form $A!B : (M)t$, then:
   (a) $t \in \overline{s_{A,m} \cup M}$ and $M \cap Sub(s) = \emptyset$;
   (b) $s'_{A,m} = s_{A,m} \cup M \cup \{t\}$, $s'_{I,m} = s_{I,m} \cup \{t\}$, and $s'_{C,m} = s_{C,m}$ for any $C \in \mathcal{A} - \{A, I\}$;
   (c) the facts in s' are obtained as follows:
      i. add $sent(A, t, B)$ to $s_{A,f}$ and $s_{I,f}$;
      ii. if some term $t_1 = \{t'\}_{K_{AC}}$ or $t_1 = \{t'\}_{K^e_C}$ has been built by A in order to build t, then add $gen(A, t_1, C)$ to $s_{A,f}$;
      iii. if some term $t_1 = (t', \{t'\}_{K^d_A})$ has been built by A in order to build t, then add $auth(A, t_1)$ to $s_{A,f}$;
      iv. $s'_{C,f} = s_{C,f}$, for any $C \in \mathcal{A} - \{A, I\}$;
2. if a is of the form $A?B : t$, then:
   (a) $t \in \overline{s_{I,m}}$;
   (b) $s'_{A,m} = s_{A,m} \cup \{t\}$ and $s'_{C,m} = s_{C,m}$, for all $C \in \mathcal{A} - \{A\}$;
   (c) the facts in s' are obtained as follows:
      i. add $rec(A, t, (B, I))$ to $s_{A,f}$ and $s_{I,f}$;
      ii. $s'_{C,f} = s_{C,f}$, for any $C \in \mathcal{A} - \{A, I\}$.

In the case of a passive intruder (2a) should be "$t \in \overline{s_{B,m}}$" and (2ci) above should be "add $rec(A, t, B)$ to $s_{A,f}$ and $s_{I,f}$". All the other concepts, such as *computation step* or *computation* (*run*), remain unchanged.

3.2 Fact Derivation

At each point in the evolution of a protocol, each agent may derive new facts from the facts he owns at that point. For instance, when A performs the first action in our running example and sends $\{A, B, H, N_A, K\}_{K^e_B}$ to B, A records the fact $sent(A, \{A, B, H, N_A, K\}_{K^e_B}, B)$ in his state. As A built this message for B, he knows all the "ingredients" he used to built it and, therefore, A may think that he sent to B each such ingredient. Therefore, from $sent(A, \{A, B, H, N_A, K\}_{K^e_B}, B)$ the agent A should be able to derive $sent(A, N_A, B)$, or $sent(A, K, B)$, and so on. Even more, A should be able to derive facts like $sent(A, N_A)$ ("A sent N_A to some agent") or $sent(A)$ ("A sent some message") or $sent(N_A, B)$ ("N_A was sent to B") or $sent(A, B)$ ("A sent some message to B") or $sent(N_A)$ ("N_A was sent by some agent"). In order not to overload the notation we have used the same predicate symbol "*sent*" to denote these new facts; the distinction will always be clear from the context (alternatively, one may use the notation $sent(A, N_A, \_)$, $sent(A, \_, \_)$, and so on).

The derivation process sketched above is guided by deduction rules. Some of these rules are based on the *trace of a message with respect to an agent state*. Intuitively, the trace of t w.r.t. $s = (s_m, s_f)$, denoted $trace(t, s)$, is the set of all messages an agent in state s could use in order to build t.

Definition 1. A message t is called *decomposable* over an agent state $s = (s_m, s_f)$ if $t \in \mathcal{T}_0$, or $t = (t_1, t_2)$ for some messages t_1 and t_2, or $t = \{t'\}_K$ for some message t' and key K with $K^{-1} \in analz(s_m)$, or $gen(A, t, B) \in s_f$ for some honest agents A and B.

"$gen(A, t, B)$" in Definition 1 covers the case when A generates t for B by encrypting some message by B's public key (A does not know B's corresponding private key but knows how he built t and, from this point of view, we may say that t is decomposable).

Definition 2. The function $trace(t, s)$, where t is a message and $s = (s_m, s_f)$ is an agent state, is given by:

- $trace(t, s) = \{t\}$, if $t \in \mathcal{T}_0$;
- $trace(t, s) = \{t\} \cup trace(t_1, s) \cup trace(t_2, s)$, if $t = (t_1, t_2)$ for some terms t_1 and t_2;
- $trace(t, s) = \{t\}$, if t is not decomposable over s;
- $trace(t, s) = \{t\} \cup trace(t', s)$, if $t = \{t'\}_K$ is an encrypted but decomposable message over s.

We are now in a position to present our deduction rules:

- *fact simplification rules*

$$(S1)\ \frac{sent(A,t,B)}{sent(A,t),sent(A,B),sent(t,B)} \qquad (S2)\ \frac{sent(A,B)}{sent(A)}$$

$$(S3)\ \frac{sent(A,t)}{sent(A),sent(t)} \qquad (S4)\ \frac{sent(t,B)}{sent(t)}$$

$$(R1)\ \frac{rec(A,t,x)}{rec(A,t),rec(A,x),rec(t,x)} \qquad (R2)\ \frac{rec(A,x)}{rec(A)}$$

$$(R3)\ \frac{rec(A,t)}{rec(A),rec(t)} \qquad (R4)\ \frac{rec(t,x)}{rec(t)}$$

  where x is B or (B, I), and B is an honest agent different than A (if "A sent t to B" then we may also say that "A send t", or "A sent some message to B", or "the message t was sent to B", and so on);
- *message simplification rules*

$$(S5)\ \frac{sent(A,t,B),\ t' \in trace(t,s)}{sent(A,t',B)} \quad (R5)\ \frac{rec(A,t,B),\ t' \in trace(t,s)}{rec(A,t',B)}$$

$$(R5')\ \frac{rec(A,t,(B,I)),\ t' \in trace(t,s)}{rec(A,t',(B,I))}$$

  where s is an agent state (if "A sent t to B" and t' was used by A to build t, then "A sent t' to B", and so on);
- *from rec-facts to gen- and auth-facts*

$$(RG)\ \frac{rec(B,\{t\}_{K_{AB}})}{gen(A,\{t\}_{K_{AB}},B)} \qquad (RA)\ \frac{rec(B,(t,\{t\}_{K_A^d}))}{auth(A,(t,\{t\}_{K_A^d}))}$$

  (if B received $\{t\}_{K_{AB}}$, then B knows that A is the only agent who could generate this message for him. If B verifies the signature on t and it turns out to be A's signature, then B knows that A authenticated the message t);

– *from rec-facts to sent-facts*

$$(RS1)\ \frac{rec(A,t,B)}{sent(B,t,A)} \qquad (RS1')\ \frac{rec(A,t,(B,I))}{sent(B)}$$

(if A knows that he received t from B, then B sent t to A; however, if A is not sure whether he received t from B, then what he knows is that B sent some message)

$$(RGS)\ \frac{rec(A,t),\ gen(C,t,A)}{sent(C,t,A)} \qquad (RAS)\ \frac{rec(A,t),\ auth(C,t)}{sent(C,t)}$$

(if A received some message t that was generated for him by C, then A can conclude that C sent the message to him. If A received an authentic message to C, then he can conclude that C sent the message);

– *from rec-facts to rec-facts*

$$(RGR)\ \frac{rec(A,t,(B,I)),\ gen(B,t,A)}{rec(A,t,B)} \qquad (RAR)\ \frac{rec(A,t,(B,I)),\ auth(B,t)}{rec(A,t,B)}$$

(if A is not sure whether he received the message t from B or from intruder, but the message t turns out to be generated by B for A or it is an authentic message to B, then A should be sure that the message t comes from B);

– *from sent-facts to sent-facts*

$$(SGS)\ \frac{sent(A,t),\ gen(A,t,B)}{sent(A,t,B)}$$

(if A sent t and generated it for B, then A sent t for B);

– *from sent-facts to rec-facts*

$$(SGR)\ \frac{sent(A,t,B),\ gen(C,t,B)}{rec(A,t,C)}$$

(if A sent t to B, and t was generated by C for B, then A received t from C).

As an example of deduction, one can easily derive from (SGR) and $(RS1)$ the following rule:

$$(RGS')\ \frac{rec(A,t,B),\ gen(C,t,A)}{sent(C,t,B)}$$

(RGS') captures a situation like the one in the Kerberos protocol (Figure 2) where C sends a ticket $\{t\}_{K_{AC}}$ to A via B. In this case, from the facts $rec(A,t,B)$ and $gen(C,t,A)$, the agent A is able to deduce $sent(C,t,A)$ (by using (RGS), $(S1)$, and (SGS)).

$$C \xrightarrow{\{\cdots,\{t\}_{K_{AC}}\}_{K_{BC}}} B \xrightarrow{\{\cdots\},\{t\}_{K_{AC}}} A$$

Fig. 2. Deduction rule (RGS')

The rule (RGS') can be used with our running example and allows H to deduce $sent(A,\{Ticket\}_{K_{AH}},C)$ at some state in the protocol (i.e., H will learn that A is the one who sent him the ticket $Ticket$).

Given a set M of messages and a set F of facts, denote by $Analz(M, F)$ the set of all facts that can be inferred from F and M. If $s = (s_m, s_f)$ is an agent state, then $Analz(s)$ stands for $Analz(s_m, s_f)$.

We note the difference between "*analz*" (Section 2) and "*Analz*".

3.3 Observational Equivalence

Anonymity, and other similar properties, are crucially based on what agents are able to "observe". If two distinct messages can be decomposed into the same atomic messages or both are encrypted by keys the agent A does not know, then the two messages are "observationally equivalent" from A's point of view in the sense that none of them reveals more "meaningful information" to A than the other. This can be extended to facts and agent states as follows.

Given a pair of agent states (s, s') define the binary relation $\sim_{s,s'}$ on message terms by:

- $t \sim_{s,s'} t$, for any $t \in \mathcal{T}_0$;
- $t \sim_{s,s'} t'$, for any term t undecomposable over s and any term t' undecomposable over s';
- $(t_1, t_2) \sim_{s,s'} (t'_1, t'_2)$, for any terms t_1, t_2, t'_1, and t'_2 with $t_1 \sim_{s,s'} t'_1$ and $t_2 \sim_{s,s'} t'_2$;
- $\{t\}_K \sim_{s,s'} \{t'\}_K$, for any terms t and t' and any key K with $t \sim_{s,s'} t'$ and $K^{-1} \in analz(s_m) \cap analz(s'_m)$.

Component-wise extend the relation $\sim_{s,s'}$ to facts:

$$P(t_1, \ldots, t_i) \sim_{s,s'} P(t'_1, \ldots, t'_i) \quad \Leftrightarrow \quad (\forall 1 \leq j \leq i)(t_j \sim_{s,s'} t'_j).$$

Definition 3. Two agent states $s = (s_m, s_f)$ and $s' = (s'_m, s'_f)$ are *observationally equivalent*, denoted $s \sim s'$, if the following hold:

- $analz(s_m) \cap \mathcal{T}_0 = analz(s'_m) \cap \mathcal{T}_0$;
- for any $\varphi \in Analz(s)$ there is $\varphi' \in Analz(s')$ such that $\varphi \sim_{s,s'} \varphi'$;
- for any $\varphi' \in Analz(s')$ there is $\varphi \in Analz(s)$ such that $\varphi' \sim_{s',s} \varphi$.

Roughly speaking, Definition 3 says that if $s = (s_m, s_f)$ and $s' = (s'_m, s'_f)$ are two observationally equivalent states of an agent, then the agent can derive the same meaningful information from any of these two states. Or, in other words, these two states are *indistinguishable*.

Let $s_m = \{\{N_C\}_K\}$, $s_f = \{rec(A, \{N_C\}_K, B)\}$, $s'_m = \{\{C, N_C\}_K\}$, and $s'_f = \{rec(A, \{C, N_C\}_K, B)\}$, where K is a symmetric key. According to Definition 3, $s = (s_m, s_f)$ and $s' = (s'_m, s'_f)$ are observationally equivalent. If we replace s_m above by $\{\{N_C\}_K, C, K\}$ and s'_m by $\{\{C, N_C\}_K, K\}$, then s and s' are not anymore observationally equivalent because from $rec(A, \{C, N_C\}_K, B)$ and s'_m one can infer $rec(A, C, B)$, and this fact cannot be inferred from $rec(A, \{N_C\}_K, B)$ and s_m.

Proposition 1. The observational equivalence on agent states is an equivalence relation decidable in $\mathcal{O}(f^4 l^4)$ time complexity, where f is the maximum number of facts in the states, and l is the maximum length of the messages in the states.

Recall that a protocol state is a tuple $s = (s_A | A \in \mathcal{A})$. We extend the equivalence relation defined above to protocol states on coordinates, that is, two protocol states s and s' are *observationally equivalent with respect to an agent* A, denoted $s \sim^A s'$ if $s_A \sim s'_A$. From Proposition 1 it follows that $\sim^A$ is an equivalence relation on protocol states, for any agent A.

3.4 Anonymity

We use the epistemic logic in [2, 7] to reason about anonymity, tailored to our paper as follows:

$$\varphi ::= p \,|\, \varphi \wedge \varphi \,|\, \neg\varphi \,|\, \mathtt{K}_A\varphi$$

where A ranges over a non-empty finite set $\mathcal{A}$ of agent names and p ranges over a set Φ of *sent*-, *rec*-, *gen*-, and *auth*-facts such that no *rec*-fact contains terms of the form (B, I).

The anonymity concepts we will define make use of only one occurrence of the operator $\mathtt{K}$ in any formula and so, the *truth value of a formula* φ in a security protocol $\mathcal{P}$ is defined inductively as follows:

- $\mathcal{P} \models \varphi$ iff $(\mathcal{P}, s) \models \varphi$, for any reachable state s in $\mathcal{P}$;
- $(\mathcal{P}, s) \models p$ iff $(\mathcal{P}, s_A) \models p$, for some agent $A \neq I$;
- $(\mathcal{P}, s_X) \models p$ iff $p \in Analz(s_X)$, where $X \in \mathcal{A}$;
- $(\mathcal{P}, s) \models \neg\varphi$ iff $(\mathcal{P}, s) \not\models \varphi$;
- $(\mathcal{P}, s) \models \varphi \wedge \psi$ iff $(\mathcal{P}, s) \models \varphi$ and $(\mathcal{P}, s) \models \psi$;
- $(\mathcal{P}, s) \models \mathtt{K}_A\varphi$ iff $(\mathcal{P}, s'_A) \models \varphi$, for any reachable state s' with $s' \sim^A s$.

The formula $\mathtt{K}_A\varphi$ means "agent A knows φ". As usual, we use $\mathtt{P}_A\varphi$ as an abbreviation for $\neg\mathtt{K}_A\neg\varphi$. $\mathtt{P}_A\varphi$ means "agent A thinks that φ is possible". We shall simply write $s \models \varphi$ instead of $(\mathcal{P}, s) \models \varphi$, whenever the protocol $\mathcal{P}$ is understood from the context.

Anonymity for security protocols will be defined for actions performed by agents. By an *action* we will understand a *sent*-fact (these are also called *sent-actions*), or a *rec*-fact that does not contain terms of the form (B, I) (these are also called *rec-actions*). Therefore, the *sent*-actions are of the form $sent(A, t, B)$, $sent(A, t)$, $sent(A, B)$, $sent(A)$, $sent(t)$, or $sent(t, B)$, while the *rec*-actions are of the form $rec(A, t, B)$, $rec(A, t)$, $rec(A, B)$, $rec(A)$, $rec(t)$, or $rec(t, B)$. By *act* we will denote a generic action of the one of the forms above.

Now, following [7], define minimal anonymity for security protocols.

Definition 4. Let $\mathcal{P}$ be a security protocol and X an agent in $\mathcal{P}$ (X may be an honest agent H or the intruder I). An action *act* of $\mathcal{P}$ is *minimally anonymous w.r.t.* X if $\mathcal{P} \models act \Rightarrow \neg\mathtt{K}_X act$.

As we can see, we have defined anonymity not only with respect to an honest agent but also with respect to the intruder. This is motivated by the fact that the intruder is an observer of the entire protocol execution and, in spite of the fact that he records all send and receive actions, he might not be able to see precisely the action performed by some agent. For instance, the intruder may be able to see that A performed a send action but he might not be able to see that A sent some specific message. On the other side, honest agents may have more deduction power than the intruder, but might not observe all send and receive actions performed in the protocol. Therefore, from the anonymity point of view, honest agents and the intruder have incomparable powers. This makes the study of anonymity with respect to the intruder very appealing.

The action $sent(B, Ticket, A)$ in our running example is minimally anonymous w.r.t. C because, whenever this action is performed, C is not able to deduce it from his knowledge. On the other side, the action $sent(A, \{Ticket\}_{K_{AH}}, C)$ is not minimally anonymous w.r.t. H because H can learn it by the deduction rule (RGS'), but it is minimally anonymous w.r.t. I because I cannot learn it.

Remark 1. We want to emphasize that the anonymity of an action which contains messages, such as $sent(A, t)$, should not be confused with the secrecy of t. The minimal anonymity of $sent(A, t)$ w.r.t. H means that H was not able to observe at some point that the agent A performed the "action of sending the message t" (although H might knew t).

Remark 2. The anonymity of an action within a group of agents (anonymity set) as defined in ([7], Definition 3.4) can be expressed in our formalism as well, and the results in Section 4 obtained for minimal anonymity hold for this kind of anonymity too. However, the lack of space does not allow us to go into details.

4 Complexity of Anonymity

In this section we establish several complexity results for the anonymity problem in security protocols. First, we fix a few notations.

Each action has a *type* which is a tuple. For instance, $sent(A, t, B)$ has type (s, a, m, a), where s stands for *sent*, a for "agent", and m for "message". Similarly, $sent(t, B)$ has type (s, m, a), $rec(A, t)$ has type (r, a, m), where r stands for *rec*, and so on.

Each action type τ induces two decision problems w.r.t. anonymity:

1. the *minimal anonymity problem for actions of type τ w.r.t. an honest agent* (abbreviated $MAP(\tau)$), which is the problem to decide, given a security protocol $\mathcal{P}$, an action *act* of type τ, and an honest agent H, whether *act* is minimally anonymous w.r.t. H in $\mathcal{P}$;
2. the *minimal anonymity problem for actions of type τ w.r.t. the intruder* (abbreviated $MAPI(\tau)$), which is the problem to decide, given a security protocol $\mathcal{P}$ and an action *act* of type τ, whether *act* is minimally anonymous w.r.t. the intruder in $\mathcal{P}$.

Minimal anonymity w.r.t. honest agents in unrestricted security protocols is undecidable. This can be obtained by reducing the halting problem for counter machines to the complement of the minimal anonymity problem. The reduction follows, somehow, a classical line for simulating counter machines [14]. When the machine halts, some action in the security protocol simulating the machine will not be minimally anonymous w.r.t. some honest agent, and this happens only when the machine halts.

Theorem 1. $MAP(\tau)$ is undecidable in unrestricted security protocols, for any action type τ.

The undecidability result in Theorem 1 can be extended to minimal anonymity w.r.t. the intruder, but not for all action types.

Theorem 2. $MAPI(\tau)$ is undecidable in unrestricted security protocols, for any action type τ except for (r, a, a), (r, m, a), and (r, a, m, a).

If we focus on bounded security protocols then the anonymity is decidable. Recall that a bounded security protocol [19] is a security protocol whose message terms are built over some finite set of basic terms and whose length do not exceed some constant k. As a conclusion, the state space of a bounded security protocol is finite and so, we should be able to decide whether an action *act* is minimally anonymous w.r.t. some agent X (honest or the intruder). An obvious algorithm for checking whether an action *act* is minimally anonymous w.r.t. X would be the following:

for *any reachable state* s *with* $s \models act$ **do**
  if *there exists a reachable state* s' *with* $s' \sim^X s$ *and* $s'_X \not\models act$ **then**
    act is minimally anonymous w.r.t. X
  else *act* is not minimally anonymous w.r.t. X
end

This algorithm searches the state space twice: once for reachable states s with $s \models act$ and then, if such a state is found, for a state s' with $s' \sim^X s$ and $s'_X \not\models act$. As the number of events of a bounded security protocol is exponential w.r.t. the size of the protocol [19], this algorithm has a very high time complexity (w.r.t. the size of the protocol).

The complexity can be cut down if we restrict the minimal anonymity problem to basic-term actions. An action *act* of a security protocol is called a *basic-term action* if all terms in the action are basic terms. For instance, $sent(A, N_A, B)$, where N_A is a nonce, is a basic-term action, whereas $sent(A, \{N_A\}_K, B)$ is not. For basic-term actions the following property holds: if $s' \sim^X s$ then $s'_X \not\models act$ if and only if $s_X \not\models act$. Therefore, for basic-term actions, the above algorithm can be simplified by replacing the test in the if-statement by the simpler one "$s_X \not\models act$". Thus, we obtain the following result.

Theorem 3. $MAP(\tau)$ and $MAPI(\tau)$ are in $NEXPTIME$ for any τ if they are restricted to basic-term actions of type τ and bounded security protocols. Moreover, except for $MAPI(r, a, a)$, $MAPI(r, m, a)$, and $MAPI(r, a, m, a)$, all

the other minimal anonymity problems restricted as above are complete for $NEXPTIME$.

If we restrict more bounded security protocols by allowing only 1-session runs, then we obtain the following complexity results.

Theorem 4. $MAP(\tau)$ and $MAPI(\tau)$ are in NP for any τ if they are restricted to basic-term actions of type τ and 1-session bounded security protocols. Moreover, except for $MAPI(r, a, a)$, $MAPI(r, m, a)$, and $MAPI(r, a, m, a)$, all the other minimal anonymity problems restricted as above are complete for NP.

5 Conclusions

Using an epistemic logic framework, we have considered in this paper a large variety of anonymity-related concepts for security protocols: six variants of sender anonymity and six variants of receiver anonymity. All of them were formulated both w.r.t. an honest agent and w.r.t. the intruder, and are based on an observational equivalence on protocol states, which is decidable in deterministic polynomial time.

We have shown that the decision problems induced by them are undecidable in unrestricted security protocols under an active intruder. For bounded (1-session bounded) security protocols we have shown that some of these decision problems are complete for NEXPTIME (NP). The status of the others is left open.

We have obtained similar results to those in Section 4 for other types of anonymity, such as the one in ([7], Definition 3.4), but they could not have been included here due to the lack of space.

References

1. Dolev, D., Yao, A.: On the Security of Public-Key Protocols. IEEE Transactions on Information Theory 29, 198–208 (1983)
2. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Reasoning About Knowledge. The MIT Press, Cambridge (2003)
3. Feigenbaum, J., Johnson, A., Syverson, P.: A model of onion routing with provable anonymity. In: Proceedings of the 11th International Conference on Financial Cryptography and 1st International Conference on Usable Security, Scarborough, Trinidad and Tobago, February 12-16 (2007)
4. Fischer, P.C., Meyer, A.R., Rosenberg, A.L.: Counter Machines and Counter Languages. Mathematical System Theory 2, 265–283 (1968)
5. Garcia, F.D., Hasuo, I., Pieters, W., van Rossum, P.: Provable Anonymity. In: Proceedings of the 3rd ACM Workshop on Formal Methods in Security Engineering: From Specifications to Code, FMSE 2005, Alexandria, USA (2005)
6. Greibach, S.A.: Remarks on Blind and Partially Blind One-way Multicounter Machines. Theoretical Computer Science 7, 311–324 (1978)
7. Halpern, J.Y., O'Neill, K.R.: Anonymity and Information Hiding in Multi-agent Systems. Journal of Computer Security 13(3), 483–514 (2005)

8. Hughes, D., Shmatikov, V.: Information Hiding, Anonymity and Privacy: A Modular Approach. Journal of Computer Security 12(1), 3–36 (2004)
9. Kramer, S.: Cryptographic Protocol Logic: Satisfaction for (Timed) Dolev-Yao Cryptography. The Journal of Logic and Algebraic Programming 77, 60–91 (2008)
10. Mano, K., Kawabe, Y., Sakurada, H., Tsukada, Y.: Role Interchangibility and Verification of Electronic Voting. In: The 2006 Symposium on Cryptography and Information Security, Hiroshima, Japan (2006)
11. Minsky, M.L.: "Recursvive" Unsolvability of Post's Problem of "Tag" and other Topics in Theory of Turing Machines. Annals of Mathematics 74(3) (1961)
12. Pfitzmann, A., Hansen, M.: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management – A Consolidated Proposal for Terminology. Technical Report, Technische Universität Dresden (2008)
13. Ramanujam, R., Suresh, S.P.: A Decidable Subclass of Unbounded Security Protocols. In: Proceedings of Workshop on Issues in the Theory of Security (WITS 2001), pp. 11–20 (2003)
14. Ramanujam, R., Suresh, S.P.: Undecidability of Secrecy for Security Protocols. Manuscript (2003) `http://www.imsc.res.in/~jam/`
15. Schneider, P., Sidiropoulos, A.: CSP and Anonymity. In: Martella, G., Kurth, H., Montolivo, E., Bertino, E. (eds.) ESORICS 1996. LNCS, vol. 1146, pp. 198–218. Springer, Heidelberg (1996)
16. Syverson, P.F., Stubblebine, S.G.: Group Principals and the Formalization of Anonymity. In: Wing, J.M., Woodcock, J.C.P., Davies, J. (eds.) FM 1999, vol. 1708, pp. 314–333. Springer, Heidelberg (1999)
17. Tsukada, Y., Mano, K., Sakurada, H., Kawabe, Y.: Anonymity, Privacy, Onymity, and Identity: A Modal Logic Approach. In: Proceedings of the 2009 IEEE International Conference on Privacy, Security, Risk and Trust (PASSAT 2009), pp. 42–51 (2009)
18. Ţiplea, F.L., Bîrjoveanu, C.V., Enea, C.: Complexity of the Secrecy for Bounded Security Protocols. In: Proceedings of the NATO Advanced Research Workshop on Information Security in Wireless Networks, Suceava, România (2006)
19. Ţiplea, F.L., Bîrjoveanu, C.V., Enea, C., Boureanu, I.: Secrecy for Bounded Protocols with Freshness Check is NEXPTIME-complete. Journal of Computer Security 16(6), 689–712 (2008)