Provable Anonymity

Flavio D. Garcia Ichiro Hasuo Wolter Pieters Peter van Rossum

Institute for Computing and Information Sciences, Radboud University Nijmegen
P.O. Box 9010, 6500 GL Nijmegen, The Netherlands
E-mail: {flaviog,ichiro,wolterp,petervr}@cs.ru.nl
URL: http://www.cs.ru.nl/{~flaviog,~ichiro,~wolterp,~petervr}

ABSTRACT

This paper provides a formal framework for the analysis of information hiding properties of anonymous communication protocols in terms of epistemic logic. The key ingredient is our notion of observational equivalence, which is based on the cryptographic structure of messages and relations between otherwise random looking messages. Two runs are considered observationally equivalent if a spy cannot discover any meaningful distinction between them. We illustrate our approach by proving sender anonymity and unlinkability for two anonymizing protocols, Onion Routing and Crowds. Moreover, we consider a version of Onion Routing in which we inject a subtle error and show how our framework is capable of capturing this flaw.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Protocol verification*; F.4 [**Mathematical Logic and Formal Languages**]

General Terms

Security, Theory, Verification

Keywords

Anonymity, Unlinkability, Privacy, Epistemic Logic, Knowledge, Cryptography, Formal Methods, Crowds, Onion Routing

1. INTRODUCTION

There is a growing concern about the amount of personal information that is being gathered about people's actions. Websites habitually track people's browsing behavior, banks track people's financial whereabouts, and current trends in applications of RFID chips will make it possible to track people's physical location. Furthermore, several European governments are considering to oblige Internet Service Providers to keep traffic logs of all communication on the Internet.

FMSE'05, November 11, 2005, Alexandria, Virginia, USA.

To protect users from undesired information leaks, there are various protocols and tools that provide some form of anonymity, e.g., for web browsing and e-mailing; for instance Tor [6], Freenet [4], and Mixminion [5].

Anonymizing Protocols In 1981, Chaum [3] pioneered the idea of what are currently called *Chaum mixes*. If A wants to send a message m to B, she chooses a number of relays (cascading mixes, in Chaum's terminology), say R_1, R_2, and sends $\{\!|R_2, \{\!|B, \{\!|m|\!\}_B|\!\}_{R_2}|\!\}_{R_1}$ to R_1. Here $\{\!|\dots|\!\}_X$ denotes encryption with the public key of X. Relay R_1 decrypts the first layer of encryption and forwards the message $\{\!|B, \{\!|m|\!\}_B|\!\}_{R_2}$ to R_2, who peels off another layer and sends the remainder to B. Traffic analysis is complicated as every relay queues messages until a certain number have been received and relays them in a batch. Also, dummy network traffic can be generated if not enough messages are available. This technique forms the basis of Onion Routing [9] and its successor Tor [6], a currently running network of relays providing anonymous communication for instance with web servers.

Another idea to provide some kind of anonymity is given by Crowds [16]. Here the message m is sent to a relay R_1, which then probabilistically either forwards the message to another relay R_2, or to its final destination. A key idea here is that every member of the network can act as a relay.

Various notions of anonymous communication exist and various attacker models have to be considered. Chaum mixes typically provide *sender anonymity*: the ultimate receiver B of the message m does not know who originated the message. It also provides *unlinkability*: someone observing all network traffic cannot tell that A and B are communicating.

Crowds does not provide unlinkability: a global eavesdropper will be able to see the message m passing through the whole network and will see the message m moving from A to B over a number of relays. However, it does provide sender anonymity even if some of the members of the network are corrupted.

Formal methods For security and authentication, a number of formal methods and automated tools have been developed to determine the correctness of security and authentication protocols (see, e.g., [14, 12, 10]). In contrast, the formalization of anonymity is still in its infancy. This paper presents a formal framework in which various information hiding properties can be expressed in a clear and precise manner. It enables us to formulate competing notions of anonymity in a uniform fashion and compare them.

Our contribution and related work Starting point of our approach, also present in [11], is the idea that various information hiding properties can best be expressed in the language of epistemic logic. This makes it possible to reason not only about the messages in a run, but also about the knowledge agents gain from these messages. For instance, sender anonymity can typically be formulated as the receiver not knowing who sent a certain message.

Central in the epistemic framework is our notion of observational equivalence of runs. An agent knows a fact of a certain run r if that fact is true in all *possible worlds*, i.e., in all runs that are observationally equivalent to r. In [11], the observational equivalence is basically assumed, whereas we actually construct such a relation.

Our notion of observational equivalence takes care of the cryptographic operations that are defined on the set of messages. We should remark that other works present similar equivalence relations on the set of messages, e.g., [13, 1]. Ours is unique in the sense that it not only deals with whether or not messages are encrypted, but also takes care of relations between messages. Especially for dealing with unlinkability this is essential. For example, if an agent A sends a message $\{\!|m|\!\}_K$ and another agent B either sends the message $\mathsf{hash}(\{\!|m|\!\}_K)$ (for instance, to acknowledge receipt of A's message), or the message $\{\!|m|\!\}_K$ (for instance, forwarding it to someone else), then a global observer could *link* A and B, even though all messages involved look like random bit strings to this global observer.

Note that this formal approach is possibilistic in nature: it only considers whether or not it is possible that A and B are communicating, but not the probability that they are. There is another body of literature studying probabilistic aspects of anonymity, e.g., [18, 17]. These papers generally *assume* the correctness of the anonymizing protocols in the formal message sense and then compute measures of anonymity.

We will show that this epistemic framework allows us to formally describe and verify various information hiding properties that are considered in the literature.

Organization of the paper We start in Section 2 with the definition of the message algebra. Then Section 3 fixes the network and intruder model and describes the fundamental notion of observational equivalence of runs. Section 4 introduces the epistemic operators and show how these operators can be used to express various information hiding properties, such as sender anonymity and unlinkability, can be expressed. Finally, to illustrate the power of our approach, we consider in Section 5 abstract versions of Crowds and Onion Routing and formally prove their anonymizing properties. We also consider a version of Onion Routing that has a subtle error and show how our framework is capable of detecting this flaw.

Notations and terminology Throughout this paper a secret key is called a *symmetric key*. In an asymmetric encryption a key pair consists of a *public key* and a *private key*.

We clearly distinguish the verbs *to know* and *to possess*: an agent *knows a fact* while it *possesses a message*. The former acts as an epistemic operator which is applied to a proposition on the run, while the latter concerns accessibility to messages via cryptographic operations such as encryption or decryption.

The set X^* consists of all the finite lists over X, that is, $X^* = \coprod_{n\in\mathbb{N}} X^n$. For a list r with length $|r|$, the i-th element of r is denoted by r_i. The head-closed sublist $\langle r_0, r_1, \ldots, r_{i-1}\rangle$ of r is denoted by $r_{<i}$.

2. THE MESSAGE ALGEBRA

In this section we present our model of the message algebra, and introduce the closure operator on a set of messages. This is done in a standard way similar to [14, 12].

Definition 1. (Message algebra) The set of messages MSG is the set given by the following BNF notation:

$$\begin{aligned}\mathsf{MSG} \ni m ::= &\, \mathsf{name}(A) \mid \mathsf{nonce}(n)\\ &|\mathsf{key}(K) \mid \mathsf{enc}(m, K)\\ &|\mathsf{hash}(m) \mid \langle m, m\rangle.\end{aligned}$$

where A is taken from a set of *agents*, n from a set of *nonces* and K from a set of *keys*. We also assume that there is a function $(-)^{-1}$ from keys to keys with the property that $(K^{-1})^{-1} = K$. If $K^{-1} = K$, then K is called a *symmetric* key, otherwise one of K and K^{-1} is called a *private* key and the other one a *public* key.

We abbreviate $\mathsf{enc}(m, K)$ to $\{\!|m|\!\}_K$ and omit angle brackets inside encryptions. Typically, an agent A has an associated public key and we denote encryption with the public key of A simply by $\{\!|m|\!\}_A$. Terms of the form $\mathsf{name}(A)$, $\mathsf{nonce}(n)$, $\mathsf{key}(K)$ are said to be *primitive*. To simplify notation, we will simply write A instead of $\mathsf{name}(A)$, and similarly for nonces and keys.

There is a natural notion of subterm relation on the set of messages, which we denote by $\preceq$. This relation is unaware of cryptographic operations; e.g., $m \preceq \{\!|m|\!\}_K$ and $m \preceq \mathsf{hash}(m)$ always hold. Furthermore, a function $\pi\colon \mathsf{MSG} \to \mathsf{MSG}$ is said to be *structure preserving* if it maps names to names, nonces to nonces, keys to keys, encrypted messages to encrypted messages, hashes to hashes, and satisfies $\pi(\langle m_1, m_2\rangle) = \langle \pi(m_1), \pi(m_2)\rangle$ for all messages m_1, m_2.
The closure $\overline{U}$ of a set U of messages consists of all the messages that can be extracted and constructed from those in U using cryptographic operations such as decryption, encryption, tupling and decomposing tuples. For the formal definition, see Appendix A. This closure operation is important here for two reasons. One reason, which is common in formal modelling of security protocols, is that an agent may send only the messages it can construct from what it has seen (see Definition 5). The other one, which is novel, is that the closure of the possessions of an agent restricts the set of runs that the agent considers to be observationally equivalent to a given run (see Definition 8).

3. RUNS AND OBSERVATIONAL EQUIVALENCE

3.1 Network model

To model anonymity properties, we have to be able to talk about the *sender* and *receiver* of a message. Therefore we include explicit sender and receiver information in our notion of *event* below. In a real world setting, an event would correspond to an IP package, which has sender and receiver information in the header and the actual message in the body.

Definition 2. (Agents, events) We denote by AG a non-empty set of agents, which has a special element spy for the intruder. An agent which is not spy is called a *regular* agent. An *event* is a triple (A, B, m) of the *sender* A, the *recipient*

B and the *delivered message* m. To make the intention clear we denote the above event by $(A \rightarrow B : m)$. The set of all events is denoted by Event.

Definition 3. (Runs) A *run* is a finite list of events, i.e. an element of the set $\mathsf{Run} := \mathsf{Event}^*$. A run describes all the events that have occurred in a network. The function $\mathsf{msg}\colon \mathsf{Run} \rightarrow \mathcal{P}(\mathsf{MSG})$ extracts all the messages occurring in a run. That is,

$$\begin{aligned}\mathsf{msg}\big(\,&(A_0 \rightarrow B_0 : m_0),\\ &(A_1 \rightarrow B_1 : m_1),\\ &\quad\vdots\\ &(A_{n-1} \rightarrow B_{n-1} : m_{n-1})\,\big) = \{m_0, m_1, \ldots, m_{n-1}\}.\end{aligned}$$

3.2 Attacker model

It is standard to verify security properties such as authentication or secrecy under the Dolev-Yao intruder model [7]. In that model the network is completely under the control of the intruder, hence no agent can be sure who sent the message it receives, or whether the intended recipient actually receives the message send.

To model anonymity properties, it is customary to use a weaker attacker model. We assume that the intruder is *passive*, i.e., *observes* all network traffic, but does not actively *modify* it. It is however possible that some regular agents are *corrupted*; we will later model this in terms of the keys the intruder possesses. This setting enables us to express that a run of a protocol satisfies a certain information hiding property as long as certain agents are not corrupted. Contrary to the intruder, the regular agents are not necessarily aware of all the events in the run; we adopt the convention that they only see the events in which they are involved as sender or receiver.

Definition 4. (Visible part of runs) Let r be a run. For a regular agent $A \in \mathsf{AG} \setminus \{\mathsf{spy}\}$ the A-*visible part* of r, denoted by $r|_A$, is the sublist of r consisting of all the events that have A in either sender or receiver field. The spy-*visible part* of r, denoted by $r|_{\mathsf{spy}}$, is identical to r.

3.3 Communication protocol

It is important to specify the set of runs *of a protocol*, which is the set of possible worlds used in the definition of knowledge. There, runs which are illegitimate with respect to a protocol are excluded because every agent knows that those runs cannot happen. In that sense, a protocol is *common knowledge*. In this paper a protocol consists of two components, namely the *candidate runs* and the *initial possessions*.

The set of candidate runs is just a set of runs. Intuitively, it consists of those runs which fulfill the requirements on which messages may/must be sent by an agent. In the examples (see Section 5) we describe the set of candidate runs by means of inference rules.

An initial possession function $\mathsf{IPo} : \mathsf{AG} \rightarrow \mathcal{P}(\mathsf{MSG})$ assigns to each agent the set of messages the agent possesses before communication takes place. Typically an agent's initial possessions consists of its private key and the public keys of all agents. We require that, for every agent A, $\mathsf{IPo}(A)$ consists of only primitive messages.

We denote by $\mathsf{Poss}_{\mathsf{IPo}}(r, A, i)$ the set of the messages that an agent A *possesses* at stage i of the run r. It is the closure of the union of: 1) A's initial possession $\mathsf{IPo}(A)$; 2) the messages A has seen up to this point, $\mathsf{msg}\big((r_{<i})\ |_A\big)$; 3) the nonces and secret keys A has freshly generated at stage i. For a formal definition, see Appendix B. The set $\mathsf{Poss}_{\mathsf{IPo}}(r, A, i)$ consists of all messages the agent A can possibly construct after have seen the first i messages of the run.

Definition 5. A run $r \in \mathsf{Run}$ is said to be *legitimate* with respect to an initial possession function $\mathsf{IPo}\colon \mathsf{AG} \rightarrow \mathcal{P}(\mathsf{MSG})$ if, for every $i \in [0, |r|-1]$, $m_i \in \mathsf{Poss}_{\mathsf{IPo}}(r, A, i)$, where $(A_i \rightarrow B_i : m_i) = r_i$.

Definition 6. (Protocols and runs of protocol) A *protocol* is a pair $(\mathsf{cr}, \mathsf{IPo})$ consisting of a set of candidate runs cr and an initial possession function IPo. A run $r \in \mathsf{Run}$ is said to be a *run of a protocol* $(\mathsf{cr}, \mathsf{IPo})$ if $r \in \mathsf{cr}$ and is legitimate with respect to the initial possession function IPo. The set of runs of a protocol $(\mathsf{cr}, \mathsf{IPo})$ is denoted by $\mathsf{Run}_{\mathsf{cr},\mathsf{IPo}}$.

Note that in the literature (e.g., [8]) a protocol is often given locally, by specifying what kind of messages an agent can send in a certain situation. For our theory, the way of specifying a protocol is unimportant; therefore we abstract from this specification by considering a given set of candidate runs.

3.4 Observational equivalence

If an agent receives an encrypted message for which it does not have the decryption key, this message looks just like a random bit string. This section formalizes this idea by defining the notion of observational equivalence. Intuitively, two sequences of messages look the same to an agent if they are the same for the messages the agent understands and if a message in one sequence looks like a random bit string to the agent, then the corresponding message in the other sequence also looks like a random bit string. Matters are slightly more complicated: we have to take care of the case where a specific random looking bit string occurs more than once in a sequence of messages; we have to take care of the possibility for someone to understand only a submessage of a message; we also have to take care of the case where certain messages look like random bit strings to the agent, but are still somehow related. For example, if the agent does not possess the symmetric key K, two messages $m_1 = \{\!|m|\!\}_K$ and $m_2 = \mathsf{hash}(\{\!|m|\!\}_K)$ both look like random bit strings. Still the agent can derive a relationship between them, namely $m_2 = \mathsf{hash}(m_1)$.

This motivates our definition of observational equivalence below. The definition is typically applied with U being equal to $\mathsf{Poss}_{\mathsf{IPo}}(r, A, |r| - 1)$, the set of messages that an agent A possesses after the run has finished.

Definition 7. (Reinterpretations of messages) Let π be a structure preserving permutation on the set MSG of messages and let $U \subseteq \mathsf{MSG}$ be a set of messages. The map π is said to be a *semi-reinterpretation under* U if it satisfies the following conditions:

$$\begin{array}{rl} \pi(p) = p & \text{for primitive terms } p\\ \pi(\{\!|m|\!\}_K) = \{\!|\pi(m)|\!\}_K & \text{if } m,\ K \text{ are in } U, \text{ or}\\ & \text{if } \{\!|m|\!\}_K,\ K^{-1} \text{ are in } U\\ \pi(\mathsf{hash}(m)) = \mathsf{hash}(\pi(m)) & \text{if } m \text{ is in } U. \end{array}$$

The map π is called a *reinterpretation under* U if it is a semi-reinterpretation under U and if π^{-1} is a semi-reinterpretation under $\pi(U)$ as well.

The above definition says that as far as an agent can observe (i.e., for all the messages in U), the permutation π preserves structural and cryptographic relationships between messages. We extend reinterpretations naturally to events (applying π to the message field of an event) and to runs (applying π to the message field of every event in a run).

LEMMA 1. *Let U be a set of messages.*

1. *The identity map on the set of messages is a reinterpretation under U.*
2. *For every reinterpretation π under U, its inverse π^{-1} is a reinterpretation under $\pi(U)$.*
3. *For all reinterpretations π under U and π' under $\pi(U)$, their composition $\pi' \circ \pi$ is a reinterpretation under U.*

PROOF. Trivial. Note that 3 is already true for semi-reinterpretations. □

Definition 8. (Observational equivalence of runs) Let $r, r' \in \mathsf{Run}_{\mathsf{cr},\mathsf{IPo}}$ be two runs of a protocol $(\mathsf{cr}, \mathsf{IPo})$ and let $A \in \mathsf{AG}$ be an agent. Two runs r and r' are said to be *observationally equivalent for an agent* A, denoted by $r \cong_A r'$, if there exists a reinterpretation π under $\mathsf{Poss}_{\mathsf{IPo}}(r, A, |r|-1)$ such that $\pi(r|_A) = r'|_A$. Such a reinterpretation will be called a *reinterpretation for* A.

LEMMA 2. *For each agent $A \in \mathsf{AG}$, the binary relation $\cong_A$ on $\mathsf{Run}_{\mathsf{cr},\mathsf{IPo}}$ is an equivalence relation.*

PROOF. This follows immediately from Lemma 1. Note that $\pi\big(\mathsf{Poss}_{\mathsf{IPo}}(r, A, |r|-1)\big) = \mathsf{Poss}_{\mathsf{IPo}}(\pi(r), A, |r|-1)\big)$ since $\mathsf{IPo}(A)$ consists solely of primitive messages and since a reinterpretation does not change the sender and receiver fields of events. □

Example 1. Consider the following two runs in which A and A' are sending the messages m and m' to B via a single relay, or Chaum mix, M. Below n and n' are fresh nonces.

r	r'
$(A \to M : \{\!\|n, B, \{\!\|m\|\!\}_B\|\!\}_M)$	$(A \to M : \{\!\|n', B, \{\!\|m'\|\!\}_B\|\!\}_M)$
$(A' \to M : \{\!\|n', B, \{\!\|m'\|\!\}_B\|\!\}_M)$	$(A' \to M : \{\!\|n, B, \{\!\|m\|\!\}_B\|\!\}_M)$
$(M \to B : \{\!\|m\|\!\}_B)$	$(M \to B : \{\!\|m\|\!\}_B)$
$(M \to B : \{\!\|m'\|\!\}_B)$	$(M \to B : \{\!\|m'\|\!\}_B)$

Assume that the spy knows the identities of all agents, all public keys and also the private key of B. Then these runs are still observationally equivalent for the spy; we can take a reinterpretation that exchanges $\{\!\|n, B, \{\!\|m\|\!\}_B\|\!\}_M$ and $\{\!\|n', B, \{\!\|m'\|\!\}_B\|\!\}_M$ and leaves $\{\!\|m\|\!\}_B$ and $\{\!\|m'\|\!\}_B$ fixed. It is important to realize that a reinterpretation π for the spy must satisfy $\pi(\{\!\|m\|\!\}_B) = \{\!\|\pi(m)\|\!\}_B$ (since B's private key is compromised), but does not necessarily have to satisfy $\pi(\{\!\|n, B, \{\!\|m\|\!\}_B\|\!\}_M) = \{\!\|n, B, \pi(\{\!\|m\|\!\}_B)\|\!\}_M$ (and similarly for m').

Without the nonces, however, the runs are not observationally equivalent.

r	r'
$(A \to M : \{\!\|B, \{\!\|m\|\!\}_B\|\!\}_M)$	$(A \to M : \{\!\|B, \{\!\|m'\|\!\}_B\|\!\}_M)$
$(A' \to M : \{\!\|B, \{\!\|m'\|\!\}_B\|\!\}_M)$	$(A' \to M : \{\!\|B, \{\!\|m\|\!\}_B\|\!\}_M)$
$(M \to B : \{\!\|m\|\!\}_B)$	$(M \to B : \{\!\|m\|\!\}_B)$
$(M \to B : \{\!\|m'\|\!\}_B)$	$(M \to B : \{\!\|m'\|\!\}_B)$

This is due to the fact that π must satisfy $\pi(\{\!\|B, \{\!\|m\|\!\}_B\|\!\}_M) = \{\!\|B, \pi(\{\!\|m\|\!\}_B)\|\!\}_M$ (and similarly for m'). Note how this nicely captures the fact that, from the viewpoint of the spy, the messages $\{\!\|B, \{\!\|m\|\!\}_B\|\!\}_M$ and $\{\!\|B, \{\!\|m'\|\!\}_B\|\!\}_M$ can indeed be distinguished: at the end of the run the spy possesses the messages $\{\!\|m\|\!\}_B$ and $\{\!\|m'\|\!\}_B$ and he can simply encrypt the messages $\langle B, \{\!\|m\|\!\}_B\rangle$ and $\langle B, \{\!\|m'\|\!\}_B\rangle$ with the public key of M. This does not hold for the version with the nonces, since the spy never possesses these nonces.

4. FORMULAS AND EPISTEMIC OPERATORS

In this section we introduce the epistemic (or modal) language as our specification language. The semantics of epistemic operators is defined in a standard way [8], taking the set $\mathsf{Run}_{\mathsf{cr},\mathsf{IPo}}$ of runs of a protocol as the set of possible worlds equipped with observational equivalence $\cong_A$. Note that our language is *semantics-based*: a formula is identified as a $\{\mathbf{T}, \mathbf{F}\}$-valued function over a model, rather than a syntactically-defined entity.

4.1 Formulas

With a formula we want to express not only a fact about a run but also that an agent knows/does not know a certain fact about a run.

Definition 9. (Formulas) A *formula* φ is a function which takes as its arguments a protocol $(\mathsf{cr}, \mathsf{IPo})$ and a run $r \in \mathsf{Run}_{\mathsf{cr},\mathsf{IPo}}$ of that protocol, and returns either $\mathbf{T}$ or $\mathbf{F}$. The set of all the formulas is denoted by Form. We follow the tradition of logic to denote the fact $\varphi(\mathsf{cr}, \mathsf{IPo}, r) = \mathbf{T}$, where $r \in \mathsf{Run}_{\mathsf{cr},\mathsf{IPo}}$, by $\mathsf{cr}, \mathsf{IPo}, r \models \varphi$. Often the protocol $(\mathsf{cr}, \mathsf{IPo})$ under consideration is clear from the context, in which case we abbreviate $\mathsf{cr}, \mathsf{IPo}, r \models \varphi$ to $r \models \varphi$. Logical connectives on formulas such as $\wedge$, $\vee$, $\to$ and $\neg$ are defined in an obvious way. A formula φ is said to be *valid* if $\mathsf{cr}, \mathsf{IPo}, r \models \varphi$ for all cr, IPo and r.

We will use the following abbreviations Sends, $\mathsf{Possesses}$, and $\mathsf{Originates}$ repeatedly in the rest of the paper. They express fundamental properties of runs.

Definition 10. The formula $A\ \mathsf{Sends}\ m\ \mathsf{to}\ B$ means: at some stage in the run, A sends a message to B which *contains* m *as a subterm.*

$$r \models A\ \mathsf{Sends}\ m\ \mathsf{to}\ B \quad \overset{\text{def}}{\Longleftrightarrow} \quad \exists i \in [0, |r|-1].\ \big(m \preceq m' \text{ where } (A \to B : m') = r_i\big).$$

We will also use $A\ \mathsf{Sends}\ m$ to mean that A sends the message m to someone.

$$r \models A\ \mathsf{Sends}\ m \quad \overset{\text{def}}{\Longleftrightarrow} \quad \exists B.\ A\ \mathsf{Sends}\ m\ \mathsf{to}\ B.$$

The formula $A\ \mathsf{Possesses}\ m$ means: *after the run has finished*, A is capable of constructing the message m.

$$r \models A\ \mathsf{Possesses}\ m \quad \overset{\text{def}}{\Longleftrightarrow} \quad m \in \mathsf{Poss}_{\mathsf{IPo}}(r, A, |r|-1).$$

The formula $A\ \mathsf{Originates}\ m$ means that: $A\ \mathsf{Sends}\ m$, but A is not relaying. More precisely, m does not appear as a subterm of a message which A has received before.

$$r \models A \text{ Originates } m \quad \overset{\text{def}}{\Longleftrightarrow}$$
$$\exists i \in [0, |r|-1].\ \exists B.\ \Big(m \preceq m' \text{ where } (A \to B : m') = r_i$$
$$\wedge\ \forall j \in [0, i-1].\ \big(m \not\preceq \hat{m} \text{ where } (A' \to A : \hat{m}) = r_j\big)\Big).$$

4.2 Epistemic operators

Using the observation equivalence relations over over the set $\mathsf{Run}_{\mathsf{cr},\mathsf{IPo}}$ of possible worlds defined in Section 3.4, we can now introduce epistemic operators in the standard way (see e.g., [8]).

Definition 11. (Epistemic operators) Let $(\mathsf{cr}, \ \mathsf{IPo})$ be a protocol. For an agent $A \in \mathsf{AG}$, the epistemic operator $\Box_A : \mathsf{Form} \to \mathsf{Form}$ is defined by:

$$\mathsf{cr}, \mathsf{IPo}, r \models \Box_A \varphi \quad \overset{\text{def}}{\Longleftrightarrow}$$
$$\forall r' \in \mathsf{Run}_{\mathsf{cr},\mathsf{IPo}}.\ \big(r' \cong_A r \implies \mathsf{cr}, \mathsf{IPo}, r' \models \varphi\big).$$

The formula $\Box_A \varphi$ is read as "after the run is completed, the agent A *knows* that φ is true". The formula $\Diamond_A \varphi$ is short for $\neg\Box_A\neg\varphi$ and read as "after the run is completed, the agent A *suspects* that φ is true".

Lemma 3 ($\Box_A$ is the **S5**-modality). *For each agent $A \in \mathsf{AG}$, the operator $\Box_A$ satisfies the following properties.*

- *(Necessitation) For each set of candidate runs* cr *and each initial possession function* IPo, *if* $\big(\forall r \in \mathsf{Run}_{\mathsf{cr},\mathsf{IPo}}.\ r \models \varphi\big)$, *then* $\big(\forall r \in \mathsf{Run}_{\mathsf{cr},\mathsf{IPo}}.\quad r \models \Box_A \varphi\big)$.
- *The following formulas are all valid: (Distribution)* $\Box_A(\varphi \to \psi) \to (\Box_A\varphi \to \Box_A\psi)$*; (T)* $\Box_A\varphi \to \varphi$*; (4)* $\Box_A\varphi \to \Box_A\Box_A\varphi$*; (5)* $\Diamond_A\varphi \to \Box_A\Diamond_A\varphi$.

In short, the operator $\Box_A$ is so-called an **S5***-modality.*

Proof. Since the epistemic operator $\Box_A$ is defined via an equivalence relation $\cong_A$ on $\mathsf{Run}_{\mathsf{cr},\mathsf{IPo}}$. See e.g., [2]. $\Box$

4.3 Expressing information hiding properties

There are a number of properties which are referred to as information hiding properties (see e.g., [15]), and a desired property might be different from one application to another. As stated in the introduction, we aim at, rather than a decisive definition of "anonymity", a framework in which we can formulate and analyze various different properties in a uniform and straightforward manner.

In this subsection we formulate some common examples of information hiding properties in our epistemic language. We use the standard notion of an *anonymity set*: it is a collection of agents among which a given agent is not identifiable. The larger this set is, the more anonymous an agent is.

Sender anonymity Suppose that r is a run of a protocol in which an agent B receives a message m. We say that r provides *sender anonymity* with anonymity set AS if it satisfies

$$r \models \bigwedge_{X \in \mathsf{AS}} \Diamond_B (X \text{ Originates } m).$$

This means that, as far as B is concerned, every agent in the anonymity set could have sent the message.

Unlinkability We say that a run r provides *unlinkability* for users A and B with anonymity set AS if

$$r \models \big(\neg\Box_{\mathsf{spy}}\varphi_0(A,B)\big) \wedge \bigwedge_{X \in \mathsf{AS}} \Diamond_{\mathsf{spy}}\varphi_0(X,B)\ ,$$

where $\varphi_0(X,Y) = \exists n.\ \big(X \text{ Sends } n \wedge Y \text{ Possesses } n\big)$. Intuitively, the left side of the conjunction means that the adversary is not certain that A sent something to B. The right side means that every other user could have sent something to B. Similarly, unlinkability between a user A and a message m could be defined as $\models \neg\Box_{\mathsf{spy}}(A \text{ Sends } m) \wedge \bigwedge_{X \in \mathsf{AS}} \Diamond_{\mathsf{spy}}(X \text{ Sends } m)$.

Plausible deniability In certain circumstances (e.g., relays), agents might be interested in showing that they did not know that they had some sensitive information m. This might be modelled by the following epistemic formula:

$$r \models \Box_{\mathsf{spy}}\neg(\Box_A(A \text{ Possesses } m))\ .$$

This formula is read as: the spy knows that A does not know that she possesses m.

5. EXAMPLES

To illustrate the applicability of the theory, we analyze the anonymity of simplified versions of Crowds and Onion Routing. For Crowds, we focus on proving sender anonymity, whereas for Onion Routing we are concerned with unlinkability.

Before starting, we need a convenient way of specifying the candidate runs of the various protocols we will be considering. We describe them with the following inference style rules:

$$\frac{e_1, \ldots, e_n}{e}\ [c] \qquad \frac{e_1, \ldots, e_n}{\overline{\overline{e}}}\ [c]$$

The first of these rules means that, under the side condition c, if a run contains events $e_1, \ldots, e_n$, then later the event e *may* appear in the run. For the second it means that e *must* appear later in the run.

For example, in a simple version of Chaum mixes, the following rule would describe the action of a router:

$$\frac{(X \to M : \{\!|Y, m|\!\}_M)}{\overline{\overline{(M \to Y : m)}}}\ [X, Y, M \text{ pairwise distinct, } M \text{ router}]$$

In normal language this means that if X sends the message $\{\!|Y, m|\!\}_M$ to M, then M must forward the message m to Y. The side condition says that this only applies if X, Y and M are three distinct agents and if M has been designated as a router.

Note that this way of specifying runs of a protocol is no integral part of the theory; it only serves as a convenient way of presenting the protocols (or, more accurately, the candidate runs) in the examples in the remainder of this section.

5.1 Crowds

The Crowds system [16] is a system for doing anonymous web transactions based on the idea that anonymity can be provided by hiding in a crowd. When someone wants to send a request to a server, she randomly selects a user from a crowd of users and asks this user to forward the request for her to the server. This user then either forwards the request to the server, or selects another random user from the crowd to do the forwarding. (Note: this only describes the request part; not the reply part). For simplicity reasons, we model requests as nonces.

Schematically, we can model this as follows:

$$\frac{}{(A \to R : \langle S, n\rangle)} \qquad \frac{(A \to R : \langle S, n\rangle)}{(R \to S : n) \text{ or } (R \to R' : \langle S, n\rangle)}$$

This obviously provides sender anonymity, in the sense that the server cannot tell from who the request really originated. In the formal framework this can be formulated as follows:

Theorem 1. *Let r be a run of Crowds in which $(A \to S : n)$ occurs with $A \neq S$. Then*

$$r \models \bigwedge_{B \in \mathsf{AS}} \Diamond_S (B \text{ Originates } n),$$

where the anonymity set AS is equal to $\mathsf{AG} \setminus \{S\}$.

Proof. Let $B \in \mathsf{AS}$ and take $r' = (B \to A : n), r$, i.e., r augmented by a fictitious event $(B \to A : n)$. Then r' is also a valid run and obviously $r' \models B$ Originates n. Because $S \neq A, B$, we have $r'|_S = r|_S$ and therefore $r' \cong_S r$. So $r \models \Diamond_S(B \text{ Originates } n)$. $\square$

Against a global eavesdropper, i.e., someone who can observe *all* network traffic, this does not provide anonymity. This is, of course, also remarked in [16].

Theorem 2. *Let r be a run of Crowds in which A freshly sends n to S over a relay R. I.e., the run contains*

$$(A \to R : \langle S, n\rangle)$$

and the nonce n is fresh in $A \to R : \langle S, n\rangle$. Then $r \models \Box_{\mathsf{spy}}(A \text{ Originates } n)$.

Proof. Suppose that $r \models \neg\Box_{\mathsf{spy}}(A \text{ Originates } n)$. Then there is a run r' satisfying $r \cong_{\mathsf{spy}} r'$ where the formula $r' \models \neg(A \text{ Originates } n)$ holds. Let π be the reinterpretation for spy such that $\pi(r) = r'$. Note that $\pi(n) = n$ and $\pi(S) = S$. Since A did not originate n in r', there must be an event $B \to A : m$ with n as subterm appearing in m before the event $A \to R : \langle S, n\rangle$. This message m must be of the form n or of the form $\langle C, n\rangle$ for some agent C, since the protocol does not allow the sending of messages of any other form and since it must have n as subterm. However, this means that $B \to A : \pi^{-1}(m)$ occurs in r before $A \to R : \langle S, n\rangle$. Note that $\pi^{-1}(m)$ is of the form $\pi^{-1}(n) = n$ or of the form $\pi^{-1}(\langle C, n\rangle) = \langle C, n\rangle$ for some agent C. This contradicts the fact that n appears freshly in the event $A \to R : \langle S, n\rangle$. $\square$

5.2 Onion Routing

Chaum mixes were already mentioned in the introduction. Users construct messages of the form $\{\!\!\{R_2, \{\!\!\{B, \{\!\!\{m\}\!\!\}_B\}\!\!\}_{R_2}\}\!\!\}_{R_1}$ (called *onions*, because of the layered encryptions) and use relays (R_1, R_2) that decrypt one layer (*peel the onion*) and forward the remainder. In this section we analyze the unlinkability of a very simple version.

An agent in the set AG belongs to one of the following families: 1) *Onion routers* who try to disguise causal relations between incoming and outgoing messages by peeling onions (i.e., removing one layer of encryption) and forwarding them.

The router collects incoming messages until it has received k messages. Then, the messages are permuted and sent in batch to their respective intended destinations. 2) *Users* who try to communicate with one another unlinkably, with the help of the onion router. 3) The *intruder* denoted by spy.

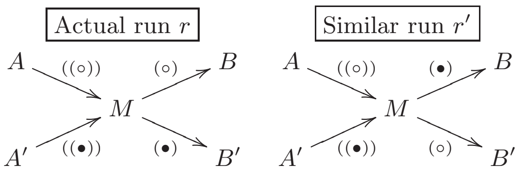

Figure 1: Onion Routing

In the sequel we assume that there is only a single router denoted by M. and furthermore the initial possession function IPo is such that, for each agent X, $\mathsf{IPo}(X)$ consists of the private key of X and the public keys of all agents. For now, no agent is corrupt, so also spy does not possess any private keys other than his own.

Again we specify the candidate runs of the protocol in the form of inference rules. The first rule says that every user X can initiate sending a message n to another user Y by building an onion and submitting it to the router M. The second says that users may also send random messages to the router M, padding the network with dummy messages. These messages will be ignored by the router; they only serve to obscure the relation between incoming and outgoing messages if not enough real traffic is available.

$$\frac{}{(X \to M : \{\!\!\{n_0, Y, \{\!\!\{n\}\!\!\}_Y\}\!\!\}_M)} \left[\begin{array}{c} X, Y, M \text{ pairwise distinct} \\ n_0, n \text{ fresh} \end{array}\right]$$

$$\frac{}{(X \to M : \{\!\!\{n\}\!\!\}_M)} \, [n \text{ fresh}]$$

The last one is

$$\frac{\begin{array}{c}(X_i \to M : \{\!\!\{n_i, Y_i, m_i\}\!\!\}_M), \\ (X_j \to M : \{\!\!\{n_j\}\!\!\}_M)\end{array}}{(M \to Y_i : m_i)} \left[\begin{array}{c} X_i, X_j, Y_i, M \\ \text{pairwise distinct} \\ i = 1 \ldots l \\ j = l+1 \ldots k \end{array}\right]$$

It means that when the router has received l onions and $k - l$ padding messages, it "peels" the onions and forwards the messages to the intended recipients. Note that l is a free variable and can be instantiated by any value $1 \leq l \leq k$, whereas k is a constant. Also note that this inference rule poses no condition on the ordering of the messages. Now we investigate under what condition onion routing ensures unlinkability, that is, which runs r of onion routing make the formula $\neg\Box_{\mathsf{spy}}\varphi_0(A, B) \wedge \bigwedge_{X \in \mathsf{AS}} \Diamond_{\mathsf{spy}}\varphi_0(X, B)$ true. Remember that we abbreviate the formula $\exists n. \big(X \text{ Sends } n \wedge Y \text{ Possesses } n\big)$ to $\varphi_0(X, Y)$.

Example 2. Consider a run r in which a user A is sending a message to a user B via the router M. Intuitively, unlinkability of A and B is ensured by showing that there is another run r' that looks similar to r (i.e., is observationally equivalent to r), but in which another user A' is actually communicating with B (and A is communicating with someone else, B'). This situation is illustrated in Figure 1. There $((\circ))$ represents a two-layered onion $\{\!\!\{n_0, B, \{\!\!\{n\}\!\!\}_B\}\!\!\}_M$. When peeled the resulting one-layered onion is denoted by

($\circ$). (($\bullet$)) is a two-layered onion with another nonce in its core.

The following theorem gives a necessary condition for a run of Onion Routing to provide unlinkability. It says that if a user A sends a message to B via the router M, then every user has to send a message to the router M to provide unlinkability of A and B.

THEOREM 3. *Let r be a run of Onion Routing which contains the events*

$$e_1 = (A \to M : \{\!|n_0, B, \{\!|n|\!\}_B|\!\}_M) \ , \quad e_2 = (M \to B : \{\!|n|\!\}_B)$$

in this order. If $r \models \neg\Box_{\mathsf{spy}}\varphi_0(A,B) \wedge \bigwedge_{X\in\mathsf{AS}} \Diamond_{\mathsf{spy}}\varphi_0(X,B)$, then r contains before e_2, for every user X, an event of the form $(X \to M : \{\!|m|\!\}_M)$ for some message m.

PROOF. Suppose we have a run r such that

$$r \models \neg\Box_{\mathsf{spy}}\varphi_0(A,B) \wedge \bigwedge_{X\in\mathsf{AS}} \Diamond_{\mathsf{spy}}\varphi_0(X,B).$$

Then $r \models \bigwedge_{X\in\mathsf{AS}} \Diamond_{\mathsf{spy}}\varphi_0(X,B)$. Now take $X \in \mathsf{AS}$. Then $r \models \Diamond_{\mathsf{spy}}\varphi_0(X,B)$, which means that there is a run r' with $r' \cong_{\mathsf{spy}} r$ such that $r' \models \varphi_0(X,B)$. Hence $r' \models X$ Sends m for some message m. Hence, r' should contain an event e of the form $(X \to M : \{\!|m'|\!\}_M)$, with $m \preceq m'$, given the inference rule for X (users can only send encrypted messages to M). Then, r must contain an event $\pi^{-1}(e')$, which finishes the proof. $\Box$

Now we give a sufficient condition for unlinkability. It says that if there are enough messages in the network, then A and B cannot be linked.

THEOREM 4. *Let r be a run of Onion Routing which contain events*

$$e = (A \to M : \{\!|n_0, B, \{\!|n|\!\}_B|\!\}_M) \ , e' = (M \to B : \{\!|n|\!\}_B)$$

in this order. Then, there exist an anonymity set AS such that $r \models \neg\Box_{\mathsf{spy}}\varphi_0(A,B) \wedge \bigwedge_{X\in\mathsf{AS}} \Diamond_{\mathsf{spy}}\varphi_0(X,B)$, with $|\mathsf{AS}| \geq k$. Recall that k is the size of the router's queue.

PROOF. We first prove the left hand side of the conjunction. That is, we want to prove that $r \models \neg\Box_{\mathsf{spy}}\varphi_0(A,B)$. This means that there should be a run r' with $r' \cong_{\mathsf{spy}} r$ such that $r' \models \neg\varphi_0(A,B)$. Let $e_1 \dots e_q$ be all the messages in r originated by A which have B as final destination, i.e., each e_i has the form $e_i = (A \to M : m_i)$ where $m_i = \{\!|\hat{n}_i, B, \{\!|n_i|\!\}_B|\!\}_M$. To keep the notation easy, we only consider the case where $q = 1$; the general case goes analogously.

By the inference rule of the router, r contains, k events $e'_j = (X_j \to M : \{\!|m_j|\!\}_M)$, $j = 1 \dots k$, one of which is e_1. These are the messages that the router M received in the same batch as e_1. Choose one of the events e'_j that is different from e_1 and define a reinterpretation π such that $\pi(m'_j) = m_1$, $\pi(m_1) = m'_j$, extending it in a natural way to the rest of the domain. Now $r' = \pi(r)$ is a run of the protocol (because the router processes messages in batches). Moreover, by the inference rule of the router, the final destinations of e'_j, $j = 1 \dots k$ are pairwise distinct and therefore $B \not\preceq \pi(m_i)$. This implies $r' \not\models \varphi_0(A,B)$.

Now we are going to prove the right side of the conjunction. Given that e' is in r, by the inference rule, there must be k events $e_i = (X_i \to M : m_i)$ with $i = 1 \dots k$ in r. Moreover, one of these events must be e, say e_j. Take $\mathsf{AS} = \{X_i \mid i = 1 \dots k\}$. Note that $|\mathsf{AS}| = k$, since the X_i are all distinct. To show that $r \models \bigwedge_{X\in\mathsf{AS}} \Diamond_{\mathsf{spy}}\varphi_0(X,B)$ part it suffices to show that for a fixed X_i, there is a run $r' = \pi(r)$ such that $r' \models \varphi_0(X_i,B)$. Now take π such that $\pi(m_i) = m_j$, $\pi(m_j) = m_i$, and the obvious extension to the rest of the domain. As before, r' is a run of the protocol and $r' \models \varphi_0(X_i,B)$. $\Box$

Finally, let us consider the case where the *private key of the router is compromised*, i.e., $\mathsf{IPo}(\mathsf{spy})$ now contains K_M^{-1}. In this case it is easy to see that unlinkability always fails.

THEOREM 5. *For any run r run of Onion Routing which contains the events*

$$e_1 = (A \to M : \{\!|n_0, B, \{\!|n|\!\}_B|\!\}_M) \ , \quad e_2 = (M \to B : \{\!|n|\!\}_B)$$

in this order, unlinkability fails. That is, $r \models \Box_{\mathsf{spy}}\varphi_0(A,B)$.

PROOF. Let r' be a run such that $r' \cong_{\mathsf{spy}} r$. Say π is a reinterpretation for spy such that $\pi(r) = r'$. Since $K_M^{-1} \in \mathsf{IPo}(\mathsf{spy})$, we have $\pi(\{\!|n_0, B, \{\!|n|\!\}_B|\!\}_M) = \{\!|n_0, B, \pi(\{\!|n|\!\}_B)|\!\}_M$. Because r' is a valid run of onion routing, $\pi(\{\!|n|\!\}_B) = \{\!|n'|\!\}_B$ for some nonce n'. Then $r' \models A$ Sends $n' \wedge B$ $\mathsf{Possesses}$ n', and therefore $r' \models \varphi_0(A,B)$. $\Box$

5.3 Onion Routing with subtle flaw

We now propose a version of Onion Routing that has a subtle error, and then show that unlinkability fails. As far as we are aware, this is the first framework that detects this kind of subtle flaws.

Imagine a modified version of Onion Routing where each channel in each path is assigned a nonce as an identifier (maybe for establishing a bi-directional communication). These nonces are reused every time that a message is sent through this channel.

The following $\langle e_0, e_1, \dots, e_7\rangle$ is an example of a run:

$$\begin{array}{ll}
e_0 = (A \to M : \langle n_{11}, ((n))\rangle) & e_4 = (A \to M : \langle n_{11}, ((n''))\rangle) \\
e_1 = (B \to M : \langle n_{21}, ((n'))\rangle) & e_5 = (B \to M : \langle n_{31}, ((n'''))\rangle) \\
e_2 = (M \to C : \langle n_{12}, (n)\rangle) & e_6 = (M \to C : \langle n_{12}, (n'')\rangle) \\
e_3 = (M \to C : \langle n_{22}, (n')\rangle) & e_7 = (M \to E : \langle n_{32}, (n''')\rangle)
\end{array}$$

where $((-))$ or $(-)$ denotes an onion like in Section 5.2, whose ultimate destination is clear from the way it is relayed. In e_2, M generates a nonce (an identifier) n_{12} and remembers the correspondence between n_{11} and n_{12}. When M relays another onion which comes with the same identifier n_{11} (in e_4), it includes the corresponding nonce n_{12} (in e_6). B uses a new identifier n_{31} in e_5 because the destination of the onion is different from e_1. From C's point of view, since the nonces n and n'' come with the same identifier n_{12}, they are from the same originator. Nevertheless the actual originator A is disguised to C.

Notice that both runs $\langle e_0, \dots, e_3\rangle$ and $\langle e_4, \dots, e_7\rangle$ ensure unlinkability in themselves. However, when combined, in the run $\langle e_0, \dots, e_7\rangle$ the users A and C are linkable, but only because the combination of identifiers n_{11} and n_{12} occurs twice in the run.

In the following we put this flaw formally. The candidate runs for Flawed Onion Routing should be clear from the above example. A user can originate an onion with an identifier, and the router M relays a peeled onion with a suitable identifier. The identifiers are either freshly generated or reused in the way illustrated in the above example.

In general, to attain unlinkability from A to B, there must be another fixed user C who immediately imitates the behavior of A: if A sends an onion bound for B, before the onion is relayed, C must send an onion bound for fixed a D. This condition is fairly unrealistic. The following theorem formally put this (unrealistic) necessary condition for unlinkability, for the case where A sends two onions.

THEOREM 6. *Let r be a run of Flawed Onion Routing which contains the events*

$$\begin{aligned} e_1 &= (A \to M : \langle n_A, \{\!|n_0, B, \{\!|n|\!\}_B|\!\}_M\rangle), \\ e_2 &= (M \to B : \langle n_M, \{\!|n|\!\}_B\rangle), \\ e_3 &= (A \to M : \langle n_A, \{\!|n_0', B, \{\!|n'|\!\}_B|\!\}_M\rangle), \\ e_4 &= (M \to B : \langle n_M, \{\!|n'|\!\}_B\rangle) \end{aligned}$$

in this order. If $r \models \neg\Box_{\mathsf{spy}}\varphi_0(A, B)$ then, r contains before e_2 and e_4, events $(X \to M : \langle n_X, m_1\rangle)$ and $(X \to M : \langle n_X, m_2\rangle)$ respectively, for some user $X \neq A$, messages m_1, m_2.

PROOF. Suppose we have a run r such that

$$r \models \neg\Box_{\mathsf{spy}}\varphi_0(A, B),$$

which means that there is a run r' such that $r' \cong_{\mathsf{spy}} r$ and $r' \models \neg\varphi_0(A, B)$. Let π be a reinterpretation for spy such that $\pi(r) = r'$. By Definition 7, r' must contain events $e_2' = \pi(e_2) = (M \to B : \langle n_M, \pi(\{\!|n|\!\}_B)\rangle)$ and $e_4' = \pi(e_4) = (M \to B : \langle n_M, \pi(\{\!|n'|\!\}_B)\rangle)$. Therefore, by the inference rule as to correspondence of identifiers, r' must contain events e_1' and e_3' of the form $e_1' = (X \to M : \langle n_X, \pi(\{\!|n_0, B, \{\!|n|\!\}_B|\!\}_M)\rangle)$ and $e_3' = (X \to M : \langle n_X, \pi(\{\!|n_0', B, \{\!|n'|\!\}_B|\!\}_M)\rangle)$, for some common sender X. Now $r' \models \neg\varphi_0(A, B)$ implies $X \neq A$, which finishes the proof. $\Box$

6. CONCLUSIONS AND FUTURE WORK

We have presented a framework for verifying a variety of information hiding properties, using modal logic. This framework provides a well defined epistemic language where various competing information hiding properties can be expressed and verified uniformly at semantical level. Our fine-grained definition of message reinterpretation captures subtleties that would not be captured in other state of the art frameworks.

Several extensions of our framework still remain. An obvious one is the consideration of a stronger adversarial model (e.g., active corrupt agents, Dolev-Yao), which should arise straightforwardly. It would also be interesting to consider applications to other fields like secure multiparty computation, and to other security properties besides information hiding, such as secrecy. From a practical perspective, proofs are slightly complicated, even with small examples. Therefore, having a tool which can perform this proofs in an automatic or semi-automatic fashion, would significantly improve the usability of the framework.

7. REFERENCES

[1] M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, 15(2):103–127, 2002.

[2] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Number 53 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2001.

[3] D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

[4] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 46–66, July 2000.

[5] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: Design of a type III anonymous remailer protocol. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, pages 2–15, May 2003.

[6] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, pages 303–320, Aug. 2004.

[7] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[8] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning About Knowledge*. MIT Press, Cambridge, MA, 1995.

[9] D. M. Goldschlag, M. G. Reed, and P. F. Syverson. Hiding Routing Information. In R. Anderson, editor, *Proceedings of Information Hiding: First International Workshop*, volume 1174 of *Lecture Notes in Computer Science*, pages 137–150. Springer-Verlag, May 1996.

[10] J. Guttman. Key compromise, strand spaces, and the authentication tests. In *Mathematical Foundations of Programming Semantics 17*, volume 47 of *Electronic Notes in Theoretical Computer Science*, pages 1–21. Elsevier, Amsterdam, 2001.

[11] J. Halpern and K. O'Neill. Anonymity and information hiding in multiagent systems. In *16th IEEE Computer Security Foundations Workshop (CSFW '03)*, pages 75–88, 2003.

[12] B. Jacobs. Semantics and logic for security protocols. Preprint, 2004.

[13] S. Mauw, J. Verschuren, and E. de Vink. A formalization of anonymity and onion routing. In *Esorics 2004*, volume 3193 of *Lecture Notes in Computer Science*, pages 109–124, Sophia Antipolis, 2004.

[14] L. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128, 1998.

[15] A. Pfitzmann and M. Köhntopp. Anonymity, unobservability, and pseudonymity: A consolidated proposal for terminology. Draft, version 0.22, July 2000.

[16] M. Reiter and A. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[17] A. Serjantov. *On the Anonymity of Anonymity Systems*. PhD thesis, University of Cambridge, Mar. 2004.

[18] V. Shmatikov. Probabilistic model checking of an anonymity system. *Journal of Computer Security*, 12(3):355–377, 2004.

APPENDIX

A. CLOSURE

For the formal definition of the closure $\overline{U}$ of a set U of messages, we use two functions $\mathsf{analz}, \mathsf{synth} : \mathcal{P}(\mathsf{MSG}) \to \mathcal{P}(\mathsf{MSG})$ which are defined much like in [14]. The *analyzing* function analz extends the input set by adding all the subterms obtained via decryption and decomposing tuples. For example if K is a symmetric key, we have

$$\mathsf{analz}\Big(\{\, K,\ \{\!\!\{\langle A, \mathsf{hash}(n)\rangle\}\!\!\}_K\,\}\Big) = \{\, K,\ \{\!\!\{\langle A, \mathsf{hash}(n)\rangle\}\!\!\}_K,\ \langle A, \mathsf{hash}(n)\rangle,\ A,\ \mathsf{hash}(n)\,\}.$$

Note that the nonce n is not included in the output. The *synthesizing* function synth outputs the set of messages which can be constructed by encryption and tupling using messages in the input set. For example if both m and the key K are in U, then $\{\!\!\{m\}\!\!\}_K$ is in $\mathsf{synth}(U)$.

Definition 12. (Analyze and synthesize) Let U be a set of messages. The *analysis* of U, notation $\mathsf{analz}(U)$, is the smallest set of messages satisfying

1. $U \subseteq \mathsf{analz}(U)$;
2. if $\{\!\!\{m\}\!\!\}_K, K^{-1} \in \mathsf{analz}(U)$, then $m \in \mathsf{analz}(U)$;
3. if $\langle m_1, m_2\rangle \in \mathsf{analz}(U)$, then $m_1, m_2 \in \mathsf{analz}(U)$.

Similarly, the *synthesis* of U, notation $\mathsf{synth}(U)$, is the smallest set of messages satisfying

1. $U \subseteq \mathsf{synth}(U)$;
2. if $m, K \in \mathsf{synth}(U)$, then $\{\!\!\{m\}\!\!\}_K \in \mathsf{synth}(U)$;
3. if $m \in \mathsf{synth}(U)$, then $\mathsf{hash}(m) \in \mathsf{synth}(U)$;
4. if $m_1, m_2 \in \mathsf{synth}(U)$, then $\langle m_1, m_2\rangle \in \mathsf{synth}(U)$.

Definition 13. (Closure) Let U be a set of messages. The set $\mathsf{synth}(\mathsf{analz}(U))$ is called the *closure* of U and is denoted by $\overline{U}$.

THEOREM 7. *The operator $U \mapsto \overline{U}$ is indeed a closure operator. That is, for any $U, V \in \mathsf{MSG}$: $U \subseteq \overline{U}$, $\overline{\overline{U}} = \overline{U}$, and $U \subseteq V \Longrightarrow \overline{U} \subseteq \overline{V}$.* □

B. POSSESSED MESSAGES

Definition 14. (Fresh nonces/keys) A nonce or key is said to be *fresh* for an initial possession function IPo, a run r and $i \in [0, |r|-1]$, if it (and its inverse, in the case of a key) does not occur in $r_{<i}$ nor in the initial possession $\mathsf{IPo}(A)$ of any agent A. The set of fresh nonces and keys for IPo, r, i is denoted by $\mathsf{Fresh}_{\mathsf{IPo}}(r, i)$.

Definition 15. (Possessed messages) Let $\mathsf{IPo} : \mathsf{AG} \to \mathcal{P}(\mathsf{MSG})$ be a initial possession function, r a run, A an agent and $i \in [0, |r|-1]$. The set of *possessed messages* for IPo, r, A at stage i, denoted by $\mathsf{Poss}_{\mathsf{IPo}}(r, A, i)$, is defined inductively on i. For every i, let r_i be $(A_i \to B_i : m_i)$. For a regular agent $A \in \mathsf{AG} \setminus \{\mathsf{spy}\}$,

$$\mathsf{Poss}_{\mathsf{IPo}}(r, A, 0) = \overline{\mathsf{IPo}(A)}\ ,$$

$$\mathsf{Poss}_{\mathsf{IPo}}(r, A, i+1) = \begin{cases} \overline{\{v \in \mathsf{Fresh}_{\mathsf{IPo}}(r, i) \mid v \preceq m_i\} \cup \mathsf{Poss}_{\mathsf{IPo}}(r, A, i)} & \\ & \text{if } A_i = A,\ (*) \\ \overline{\{m_i\} \cup \mathsf{Poss}_{\mathsf{IPo}}(r, A, i)} & \text{if } B_i = A, \\ \mathsf{Poss}_{\mathsf{IPo}}(r, A, i) & \text{otherwise.} \end{cases}$$

For the intruder spy,

$$\mathsf{Poss}_{\mathsf{IPo}}(r, \mathsf{spy}, 0) = \overline{\mathsf{IPo}(\mathsf{spy})}\ ,$$

$$\mathsf{Poss}_{\mathsf{IPo}}(r, \mathsf{spy}, i+1) = \begin{cases} \overline{\{v \in \mathsf{Fresh}_{\mathsf{IPo}}(r, i) \mid v \preceq m_i\} \cup \mathsf{Poss}_{\mathsf{IPo}}(r, \mathsf{spy}, i)} & \\ & \text{if } A_i = \mathsf{spy},\ (*) \\ \overline{\{m_i\} \cup \mathsf{Poss}_{\mathsf{IPo}}(r, \mathsf{spy}, i)} & \text{otherwise.} \end{cases}$$

This definition says that if fresh nonces/keys occur as subterms in a sent message (i.e., if the sender freshly generates nonces/keys), then these generated terms are added to the sender's possessed messages. This complicated definition can be understood with the following example. If an agent A sends at stage i the public-key encryption $\{\!\!\{n\}\!\!\}_B$ of a freshly generated nonce n, A is supposed to possess the nonce n (that A has generated herself) from then on. However, without the above clause $(*)$, when A does not possess the private key, the nonce n would not be in $\mathsf{Poss}_{\mathsf{IPo}}(r, A, i+1)$.

When an agent sees a message (whether as the receiver or as an eavesdropper), the message is added to the ingredients the agent can use to construct messages. For example, if an agent receives at stage i an encryption of a nonce $\{\!\!\{n\}\!\!\}_K$ which the agent cannot decrypt at that time, and later at stage $j(> i)$ the agent receives the decryption key K^{-1}, then from stage $j+1$ on the agent can use the nonce n to construct messages it sends.

The definition for spy differs from that for a regular agent because spy is a global eavesdropper and hence sees every message in the network.