An Inductive Approach to Provable Anonymity

Yongjian Li
The State Key Laboratory of Computer Sciences
Institute of Software
Chinese Academy of Sciences

Jun Pang
Computer Science and Communications
University of Luxembourg

Abstract—We formalise in a theorem prover the notion of provable anonymity proposed by Garcia et al. Our formalization relies on inductive definitions of message distinguish ability and observational equivalence over observed traces by the intruder. Our theory differs from its original proposal which essentially boils down to the existence of a reinterpretation function. We build our theory in Isabelle/HOL to have a mechanical framework for the analysis of anonymity protocols. Its feasibility is illustrated through the onion routing protocol.

I. INTRODUCTION

With the rapid growth of the Internet community and the rapid advances in technology over the past decades, people are getting used to carry out their daily activities through networked distributed systems providing electronic services to users. In these systems, people become more and more concerned about their privacy and how their personal information has been used. Typically, anonymity is a desired property of such systems, referring to the ability of a user to own some data or take some actions without being tracked down. This property is essential in systems that might involve sensitive personal data, like electronic auctions, voting, anonymous broadcasts, file-sharing etc. For example, users want to keep anonymous when they visit a particular web site or post their political opinions on a public bulletin board.

Due to its subtle nature, anonymity has been the subject of many theoretical studies and formal verification [1], [2], [3], [4], [5]. The proposed definitions aim to capture different aspects of anonymity (either probabilistic or deterministic) formal verification treats systems in different application domains, such as electronic voting systems, electronic cash protocols file sharing. However, automatic approaches to the formal verification of anonymity have mostly focused on the model checking approach on systems with fixed configurations [1], while theorem proving is a more suitable approach when dealing with general systems of infinite state spaces. We address this situation by investigating the possibility of using a powerful general-purpose theorem prover, Isabelle/HOL [6], to semi-automatically verify anonymity properties.

We start by formalising the notion of provable anonymity proposed by Garcia et al. [2]. Their key idea is to define observational equivalence between protocol traces. Two traces are to be considered equivalent if an intruder cannot distinguish them, i.e., he cannot find any meaningful difference. The distinguishing ability of the intruder is formalised as the ability to distinguish two messages, which is in turn based on message structures and relations between random looking messages. Central to their framework is the reinterpretation function proposed by Garcia et al. [2]. Proving two traces equivalent essentially boils down to the existence of such a reinterpretation function. Within their framework, Garcia et al. also define epistemic operators and use them to express information hiding properties like sender anonymity and un-linkability.

Our contribution: Our formalization of observational equivalence between traces relies on a definition of message distinguishability. Observational equivalence of traces is in the center of the epistemic framework – an agent knows a fact of a certain trace if that fact is true in all traces that are observationally equivalent to that trace. We build our theory in Isabelle/HOL [6] to have a mechanical framework for the analysis of anonymity protocols. We illustrate the feasibility of the mechanical framework through the onion routing protocol [7]. We inductively define the semantics of an onion routing protocol as a set of traces, and the relaying mechanism of the protocol is formally defined as a set of inductive rules. Furthermore, we formally prove that the protocol realizes anonymity properties such as sender anonymity and unlinkability under some circumstance by providing a method to construct an observationally equivalent onion trace for a given trace. To the best of our knowledge, theory of anonymity has not been formalised in a theorem prover yet. Our work aims to to bridge this gap. All lemmas in the paper are proved semi-automatically in Isabelle/HOL. Proofs are mostly omitted for the sake of brevity.

II. PRELIMINARIES

A. Agents, messages and events

Agents send or receive messages. There are three kinds of agents: the server, the friendly agents, and the spy. Formally the type of agent is defined as follows:

\[ \text{agent ::= Server | Friend } N | \text{Spy} \]

We use bad to denote the set of intruders, which at least includes the agent Spy. If an agent \( A \) is not in bad, then \( A \) is honest.

The set of messages is defined using the following BNF notation:

\[ h ::= \text{Agent } A | \text{Nonce } N | \text{Key } K | \text{MPair } h_1 h_2 | \text{Crypt } K h \]
where \( A \) is an element from agents, \( N \) from natural numbers, and \( K \) from natural numbers. Here we use \( K^{-1} \) to denote the inverse key of \( K \). MPair \( h_1 h_2 \) is called a composed message. Crypt \( K h \) represents the encryption of message \( h \) with \( K \).

In an asymmetric key protocol model, an agent \( A \) has a public key \( pubK A \), which is known to all agents, and a private key \( priK A \).pub \( K A \) is the inverse key of \( priK A \), and vice versa. In a symmetric key model, each agent \( A \) has a long-term symmetric key \( shrK A \). The inverse key of \( shrK A \) is itself.

Two operators \texttt{parts} and \texttt{analz} are inductively defined on a message set \( H \). Their definition is taken from [8] and tailored for our purposes. Usually, \( H \) contains a penetrator’s initial knowledge and all messages sent by regular agents. The set \( \text{parts} \) is obtained from \( H \) by repeatedly adding the components of compound messages and the bodies of encrypted messages. Formally, \( \text{parts} \) is the least set including \( H \) and closed under projection and decryption.

The \texttt{parts} operator can be used to define the subterm relation \( \sqsubseteq: h_1 \sqsubseteq h_2 \equiv h_1 \subseteq \text{parts}\{h_2\} \). Here \( K \) is not regarded as occurring in \( \{g\}_{K} \) unless \( K \) is a part of \( g \).

Similarly, \( \text{analz} \) \( H \) is defined to be the least set including \( H \) and closed under projection and decryption by known keys.

A protocol’s behaviour is specified as the set of possible traces of events. A trace model is concrete and easy to explain. An event is of the form: Says \( A B m \), which means that \( A \) send \( B \) the message \( m \). For an event \( ev = \text{Says} \ A B m \), we define \( \text{msgPart} ev \equiv m \), \( \text{sender} ev \equiv A \), \( \text{receiver} ev \equiv B \) to represent the message, sender and receiver of \( ev \). Function \( \text{initState} \) \( A \) specifies agent \( A \)’s initial knowledge. Typically an agent’s initial knowledge consists of its private key and the public keys of all agents.

The function \( \text{knows} \ A tr \) describes the set of messages which \( A \) can observe from the trace \( tr \) in addition to his initial knowledge. Formally,

\[
\text{knows} \ A [\land \text{initState} \ A] \\
\text{knows} \ A ((\text{Says} \ A' B' m)\#evs) = \\
\quad \text{if} \ (A' = \text{Spy}) \lor (A' = A) \lor (A = B) \\
\quad \text{then} \ [m] \cup \text{knows} \ A \ evs \\
\quad \text{else} \ \text{knows} \ A \ evs
\]

The set \( \text{used} evs \) formalises the notion of freshness. The set includes the set of the parts of the messages sent in the network as well as all messages held initially by any agent.

\[
\text{used} [\lor B. \ \text{parts} \ (\text{initState} B)] \\
\text{used} ((\text{Says} \ A B m)\#evs) = \text{parts}(m) \cup \text{used} \ evs
\]

Function \( \text{noncesOf} \) \( \text{msg} \equiv \{m.\exists n. m \sqsubseteq \text{msg} \land m = \text{Nonce} n\} \) defines the set of nonces occurring in the message \( \text{msg} \). The formula \( \text{originates} \ A m \ tr \) means that \( A \) originates a fresh message \( m \) in the trace \( tr \). Formally,

\[
\text{originates} \ A m [\land \text{initState} \ A] \\
\text{originates} \ A m ((\text{Says} \ A' B' \ msg)\#evs) = \\
\quad \text{if} \ (\text{originates} \ A m \ evs) \\
\quad \quad \text{then} \ True \\
\quad \quad \text{else} \ \text{if} \ (m \sqsubseteq \text{msg} \land A = A') \text{ then} \ True \\
\quad \quad \text{else} \ False
\]

The predicate \( \text{send} \ A m m \ tr \) means that \( A \) sends a message \( m \) in an event of the trace \( tr \). Formally,

\[
\text{send} \ A m m [\land \text{initState} \ A] \\
\text{send} \ A m ((\text{Says} \ A' B' m)\#evs) = \\
\quad \text{if} \ (m \sqsubseteq \text{msg} \land A = A') \text{ then} \ True \\
\quad \quad \text{else} \ \text{send} \ A m \ evs
\]

The predicate \( \text{regularOrig} m \ tr \) is to define a message originated by an honest agent. Formally, \( \text{regularOrig} m \ tr \equiv \forall A. \text{originates} \ A m \ tr \rightarrow A \notin \text{bad} \).

Next we define a set of special lists: \( \text{mutualDiffL} \). If \( L \in \text{mutualDiffL} \), \( i, j < \text{length} \ L \), and \( i \neq j \), then we have \( L_i \neq L_j \). Here \( L_i \) is the \( i \)-th element of the list \( L \).

\[
\text{inductive set} \ \text{mutualDiffL}::\{\text{'a list}\} \ \text{set where} \\
\text{nilDiff}: "[] \in \text{mutualDiffL}" \\
\text{consDiff}: "[L \in \text{mutualDiffL}, \\
\forall l. l \in (\text{set} L) \rightarrow l \neq a \implies (a \# L) \in \text{mutualDiffL}")
\]

We define single\_valued \( R \) as \( \forall x y. (x, y) \in R \rightarrow (\forall z. (x, z) \in R \rightarrow y = z) \). Obviously, if \( L \in \text{mutualDiffL} \), then single\_valued \( \text{zip} \ L \ L' \) for any \( L' \).

B. Intruder model

We discuss anonymity properties based on observations of the intruder. In this section, we explain our intruder model. Dolev-Yao intruder model [9] is considered standard in the field of formal symbolic analysis of authentication or secrecy properties of security protocols. In this model the network is completely under the control of the intruder: all messages sent on the network are read by the intruder; all received messages on the network are created or forwarded by the intruder; the intruder can also remove messages from the network. However, in the analysis of anonymity protocols, we would like to adapt a weaker attacker model. We assume that the intruder is passive in the sense that he observes all network traffic, but does not actively modify the messages or inject new messages. He can analyze the messages he has observed, which is modelled by the operator \( \text{analz} \). In later section, we will point out that some anonymity properties cannot be kept if we have the Dolev-Yao intruder model.

III. MESSAGE DISTINGUISHABILITY

In this section, we focus on modelling the ability for the agent to distinguish two received messages based on his knowledge. In principle, an agent can uniquely identify any plain-text message he observes. Furthermore, an agent can distinguish any encrypted message for which he possesses the decryption key, or which he can construct himself. Formally, if \( m \) and \( m' \) are of different type of messages, for instance, if \( m = \text{Agent} A \) and \( m' = \text{Nonce} n \), the agent can immediately tell the difference; if \( m = \{g\}_{K_1} \) and \( n = \{h\}_{K_2} \), then the agent must use the knowledge \( \text{Know} \) he possesses to decide whether the two messages are different. There are cases as shown below:

- Both \( k_1 \) and \( k_2 \) are in \( \text{Know} \), \( g, h \) are in \( \text{Know} \) as well, and the agent can distinguished \( g \) and \( h \), then he also can tell the difference between \( m \) and \( m' \) as he knows that \( m \) and \( m' \) are different encrypted messages;
IV. OBSERVATIONAL EQUIVALENCE

We first introduce the notion of observational equivalence between messages which is naturally defined as the negation of message distinguishability. If an agent cannot distinguish two messages \(m \) and \(m'\), then the two messages are observationally equivalent to the agent.

\[
\text{msgEq}:: \text{"msg set}\Rightarrow \text{"msg x msg set"}
\]

\[
\text{msgEq} \equiv \text{msg} \neq \text{msg}
\]

Obviously, observational equivalence between messages w.r.t. a knowledge set \(\text{Knows}\) is reflexive, symmetric.

Intuitively, we can lift observational equivalence to traces: two sequences of messages in two traces look the same to an agent if they are the same for the messages the agent understands and if a message in one sequence is observationally equivalent to the corresponding message in the other sequence w.r.t. the knowledge which the agent has obtained from the two traces. Besides the requirement of message matching, we also require that the sender and receiver of an event in a trace is the same as those of the corresponding event in the other sequence. For events \(e_1\) and \(e_2\), we define \(\text{SRMatch} e_1 \equiv \text{SRMatch} e_2\) means that for any trace \(tr \equiv \text{SRMatch} tr'\), the senders and receivers of \(tr\) are the same as those of \(tr'\).

Furthermore, \(\text{msgPart} tr\) and \(\text{msgPart} tr'\) are observationally equivalent to each other w.r.t. the knowledge obtained after observing the two traces. At last single_value \(H\) and single_value \(H\) guarantee that an agent cannot reinterpret an event differently.

V. EPISTEMIC OPERATORS AND ANONYMITY PROPERTIES

Using the observational equivalence relations over a trace set of possible worlds, we can formally introduce epistemic operators [2] as follows:

\[
\text{box}:: \text{"agent}\Rightarrow \text{"trace}\Rightarrow \text{"trace}\Rightarrow \text{"bool"}
\]

\[
\text{assertOfTrace}:: \text{"assertion}\Rightarrow \text{"trace set"}
\]

\[
\sigma x y z: = (x,y)\in H \implies \text{msgEq} x y
\]

\[
\forall x,y: (x,y)\in H \implies \text{msgEq} x y
\]

For notational convenience, we write \(r \equiv \Diamond A \text{~}\) for box \(A \text{~}\) and \(r \equiv A \text{~}\) for diamond \(A \text{~}\). Note that \(\Diamond \) is a predicate on a trace. Intuitively, \(r \equiv \Diamond A \text{~}\) means that for any trace \(r'\) in \(rs\), if \(r'\) is observationally equivalent to \(r\) for agent \(A\), then \(r'\) satisfies the assertion \(\varphi\). On the other hand, \(r \equiv A \text{~}\) means that there is a trace \(r'\) in \(rs\), \(r'\) is observationally equivalent to \(r\) for agent \(A\) and \(r'\) satisfies the assertion \(\varphi\). Now we can formulate some 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.

A. Sender anonymity

Suppose that \(r\) is a trace of a protocol in which a message \(m\) is originated by some agent. We say that \(r\) provides sender
Anonymity with anonymity set \( AS \) w.r.t a set of possible runs in the view of \( B \) if it satisfies:

\[
\text{constdefs senderAnonimity::"agent set\Rightarrow agent\Rightarrow msg\Rightarrow trace\Rightarrow trace set\Rightarrow bool"}
\text{senderAnonimity AS B m r rs = ((X\sqcup X)\Rightarrow AS \Rightarrow r \Rightarrow \sqcup B \Rightarrow rs \Rightarrow (originates X m)).}
\]

Here, \( AS \) is the set of agents who are under consideration, and \( rs \) is the set of all the traces which \( B \) can observe. Intuitively, this definition means that each agent in \( AS \) can originate \( m \) in a trace of \( rs \). Therefore, this means that \( B \) cannot be sure of anyone who originates this message.

**B. Unlinkability**

We say that a trace \( r \) provides unlinkability for user \( A \) and a message \( m \) w.r.t anonymity set \( AS \) if

\[
\text{constdefs unlinkability::"agent set\Rightarrow agent\Rightarrow msg\Rightarrow trace\Rightarrow trace set\Rightarrow bool"}
\text{unlinkability AS A m r rs = (let P = AX m' \Rightarrow \text{r } \Rightarrow \sqcup \text{Spy rs (P A m)) \wedge (X\sqcup X)\Rightarrow AS \Rightarrow \text{r } \Rightarrow \sqcup B \Rightarrow rs \Rightarrow (originates X m)).}
\]

where the left side of the conjunction means that the intruder is not certain that \( A \) sent \( m \), while the right side means that every other user could have sent \( m \).

**VI. CASE STUDY: ONION ROUTING PROTOCOL**

**A. Modeling the protocol**

In our work, we model a simplified onion routing protocol system, composed of a user set \( AS \) and a router \( M \), with \( M \notin AS \). We also assume that each agent can send a message before the router \( M \) launches a batch of forwarding process, and the router does not accept any message when it is forwarding messages.

\[
\text{inductive set oneOnionSession::"nat\Rightarrow agent\Rightarrow trace set for k::"nat" and M::"agent" where}
\text{onionNil1: "[] \in (oneOnionSession k M) \&
\text{onionCons1: "}\left\{ [\text{tr}\in (oneOnionSession k M) \mid X\notin M; Y\notin M; \text{Nonce n0}(\text{used tr}); \text{Nonce n0}(\text{used tr}); \text{length tr}<k] \Rightarrow
\text{\text{Says X M (Crypt (pubK M)) \& \{Nonce n0, Agent Y, (Nonce n0)\}}
\text{\text{oneOnionSession k M\}}
\text{\text{onionCons2: }\left\{ [\text{tr}\in (oneOnionSession k M) \mid X\notin M; \text{Nonce n0}(\text{used tr}); \text{length tr}<k] \Rightarrow
\text{\text{Says X M (Crypt (pubK M)) \& \{Nonce n0\}}
\text{\text{oneOnionSession k M\}}
\text{\text{onionCons3: }\left\{ [\text{tr}\in (oneOnionSession k M) \mid \text{length tr}>2k; \text{Says M Y (Crypt (pubK Y)) \& \{Nonce n0\}} \& (set tr)] \Rightarrow
\text{\text{Says M Y (Crypt (pubK Y)) \& \{Nonce n0\}}.
\text{\text{oneOnionSession k M\}}
\text{\text{oneOnionSession k M\}}}
\]}
\]

In this definition, there are four induction rules. Rule \text{Nil} specifies an empty trace. The other rules specify trace’s extension with protocol steps. The ideas behind the other induction rules are illustrated as follows. More precisely,

- If the length of the current trace is less than \( k \), namely, \( M \) is still in a receiving status, \( X \) (or \( Y \)) and \( M \) are distinct, and both \( n0 \) and \( n \) are fresh, then we can add an event \( \text{Says X M } \{ \text{Nonce n0, Agent Y, (Nonce n0)} \} \| \text{pubK M} \). This step means that \( X \) sends a message to \( M \) which will be peeled and forwarded to \( Y \) by \( M \).
- If the length of the current trace is less than \( k \), \( X \) and \( M \) are distinct, and \( n \) is fresh, then we can add an event \( \text{Says X M } \{ \text{Nonce n0, Agent Y, (Nonce n0)} \}_\| \text{pubK Y} \| \text{pubK M} \). This means that \( X \) sends a dummy message to \( M \) which will be simply discarded later.
- If the length of the current trace is greater than or equal to \( k \), namely, \( M \) is in a forwarding status, a message \( \{ \text{Nonce n0, Agent Y, (Nonce n0)} \}_\| \text{pubK Y} \| \text{pubK M} \) has been received by the router, but the peeled onion \( \{ \text{Nonce n0} \}_\| \text{pubK Y} \| \text{pubK M} \) has not been forwarded, then we can add an event \( \text{Says M Y } \{ \text{Nonce n0} \}_\| \text{pubK Y} \). This step means that the router \( M \) forwards the peeled message to \( Y \).

**B. Properties on protocol sessions**

As mentioned in a previous section, whether two traces are observationally equivalent for an agent depends on the knowledge of the agent after his observation of the two traces. Therefore, we need to discuss some properties on the knowledge of the intruder. They are secrecy properties, and some regularity on the correspondence of the events in one protocol session.

1) **Secrecy properties:** If the router \( M \) is honest, \( B \) is also honest, and \( B \) sends a message \( \{ \text{Nonce n0, Agent Y, (Nonce n0)} \}_\| \text{pubK Y} \| \text{pubK M} \) to \( M \), either \( n0 \neq n \) or \( Y \notin \text{bad} \), then \( \text{Nonce n0} \) cannot be analyzed by the intruder.

**Lemma 1**

\[
[tr \in \text{oneOnionSession } k M; n0 \neq n \vee Y \notin \text{bad}; \\
\text{Says B M } \{ \text{Nonce n0, Agent Y, (Nonce n0)} \}_\| \text{pubK Y} \| \text{pubK M} \in tr; \\
M \notin \text{bad} \vee B \notin \text{bad} ] \Rightarrow \text{Nonce n0} \notin \text{analz (spies evs)}
\]

Provided that both \( M \) and \( B \) are honest, and \( B \) sends a dummy message \( \{ \text{Nonce n0} \}_\| \text{pubK M} \) to \( M \), then \( \text{Nonce n0} \) cannot be analyzed by the intruder.

**Lemma 2**

\[
[tr \in \text{oneOnionSession } k M; \\
\text{Says B M } \{ \text{Nonce n0} \}_\| \text{pubK M} \in tr; \\
M \notin \text{bad} \vee B \notin \text{bad} ] \Rightarrow \text{Nonce n0} \notin \text{analz (spies evs)}
\]

2) **Correspondence properties:** The following lemma is about the correspondence of two events in a trace \( tr \). If the router \( M \) forwards a message \( \{ \text{Nonce n} \}_\| \text{pubK Y} \), then there must exist an agent \( A \) who has sent a message for some nonce \( n0 \) \( \{ \text{Nonce n0, Agent Y, (Nonce n0)} \}_\| \text{pubK Y} \| \text{pubK M} \).

**Lemma 3**

\[
[tr \in \text{oneOnionSession } k M; \\
\text{ma} = \{ \text{Nonce n}_\| \text{pubK Y}; \\
\text{Says M B ma} \in \text{set tr}; ] \Rightarrow \\
\exists n0 A \text{Says A M } \{ \text{Nonce n0, Agent Y, (Nonce n0)} \}_\| \text{pubK Y} \| \text{pubK M} \in \\
\text{set tr}
\]
Lemma 4
\[ tr \in \text{oneOnionSession} \ k \ M; \]
\[ ma' = \{ \text{Nonce } n \}_\text{pubK} \ ; \text{Says } A \ M \ ma \in \set tr; ma' \cap ma \} \implies \text{originates } A \ ma' \ tr \]

For a trace \( tr \in \text{oneOnionSession} \ k \ M \), an agent \( A \) sends
the router \( M \) a message \( m \), then \( A \) is not the router \( M \).

Lemma 5
\[ tr \in \text{oneOnionSession} \ k \ M; \text{Says } A \ M \ m \in \set tr \] \implies \( A \neq M \)

c) Uniqueness properties.: Since an agent is required to
originate fresh nonces when he sends a message to the router,
therefore if two events where agents send a message to the
router \( M \), either two events are exactly the same, or nonces
used in the two events are disjoint.

Lemma 6
\[ tr \in \text{oneOnionSession} \ k \ M; \text{Says } X \ M \ ma; \text{Says } Y \ M \ mb \] \implies \((X = Y \land ma = mb) \lor \)
\((\text{noncesOf } ma) \cap (\text{noncesOf } mb) = \emptyset \)

From Lemma 6, we can easily derive that once a nonce \( n \)
occurring in a message sent by an agent \( X \), then another agent
\( Y \) cannot send a message containing the same nonce \( n \).

Lemma 7
\[ tr \in \text{oneOnionSession} \ k \ M; \text{Says } X \ M \ ma; \]
\( X \neq Y; \text{Nonce } n \cap ma \] \implies \( \neg \text{sends } Y \text{ (Nonce } n \text{)} \ tr \)

The message of each event in a trace of the protocol is
unique, namely two messages in two events in this trace are
different.

Lemma 8
\[ tr \in \text{oneOnionSession} \ k \ M \] \implies \( \text{map msgPart } tr \in \text{mutualDiffL} \)

\( (\text{zip } (\text{map msgPart } tr) \ L) \) must be \text{single_valued} if \( tr \) is
in a trace of the onion routing protocol.

Lemma 9
\[ tr \in \text{oneOnionSession} \ k \ M \]
\implies \( \text{single_valued } (\text{zip } (\text{map msgPart } tr) \ L) \)

C. Traces swapping two messages

By definition of sender anonymity, the proof strategy of
such property is roughly as follows: fix an agent \( X \), we need
to prove the existence of an observationally equivalent trace
\( tr' \) w.r.t. a given trace \( tr \), where both \( tr \) and \( tr' \) are some
protocol sessions. Obviously, this means a construction of an
observationally equivalent trace \( tr' \). In this section, we discuss
this construction method in details.

We define a function \( \text{swap } ma mb tr \), which returns another
trace \( tr' \) satisfying that the sender and receiver of any event
\( tr'_i \) is the same as those in \( tr_i \), but the sent message of \( tr'_i \) is
swapped as \( mb \) if that of \( tr_i \) is \( ma \), and as \( ma \) if that of \( tr_i \)
is \( mb \), otherwise it is kept the same as that of \( tr_i \).

![Fig. 1. An illustration of function swap.](image)

For a trace \( tr \) of the onion routing protocol, Fig. 1 il-
lustrates the correspondence between \( tr \) and the function
\( \text{swap } ma mb tr \). In session 1, agent \( A \) \((B)\) communicates
with \( C \) \((D)\), while agent \( A \) \((B)\) communicates with \( D \) \((C)\)
in session 2. The correspondence between \( tr \) and \( \text{swap } ma mb tr \)
is formalised as the following lemma.

Lemma 10 Let \( tr \) be a trace.
1) \( \{ (m_1, m_2) \in \set (\text{zip } (\text{map msgPart } tr)) \} \implies m_1 = m_2 \lor \)
\((m_1, m_2) = (m_a, mb) \lor (m_1, m_2) = (mb, m_a) \)
2) \( \text{sendRecvMatch} \ tr \ (\text{swap } ma mb tr) \)
3) \( \text{length } (\text{swap } ma mb tr) = \text{length } tr \)
4) \( \text{swap } ma mb tr = \text{swap } mb ma tr \)
5) \( \{ (\text{Says } X \ M \ ma \in \set tr) \} \]
\implies \( \text{Says } X \ M \ mb \in \set (\text{swap } ma mb tr) \)
6) \( \{ (\text{Says } X \ M \ mb \in \set tr) \} \]
\implies \( \text{Says } X \ M \ ma \in \set (\text{swap } ma mb tr) \)
7) \( \{ m \neq ma; m \neq mb; (\text{Says } X \ M \ m) \in \set tr \} \]
\implies \( (\text{Says } X \ M \ m \in \set (\text{swap } ma mb tr) \)
8) \( \{ m \neq ma; m \neq mb; (\text{Says } X \ M \ m) \notin \set tr \} \]
\implies \( (\text{Says } X \ M \ m \notin \set (\text{swap } ma mb tr) \)
9) \( \{ \text{Says } A \ M \ ma \in \set tr; \text{Says } B \ M \ mb \in \set tr; A \neq \)
\text{Spy}; B \neq \text{Spy} \}
\implies \( \text{knows } \text{Spy } tr = \text{knows } \text{Spy } (\text{swap } ma mb tr) \)

Based on the lemma 10, we can conclude an important
result: for a trace \( tr \in \text{oneOnionSession} \ k \ M \), both \( ma \) and
\( mb \) are sent to the router \( M \) by some agents in \( tr \), then
\( \text{swap } ma mb tr \) is still in \( \text{oneOnionSession} \ k \ M \).

Theorem 1
\[ tr \in \text{oneOnionSession} \ k \ M; \text{Says } A \ M \ ma \in \]
\( tr; \text{Says } B \ M \ mb \in \set tr \] \implies \( \text{swap } ma mb tr \in \text{oneOnionSession} \ k \ M \)
If \( ma = \{\text{Nonce } n_0, \text{Agent } Y, \{\text{Nonce } n\}_{\text{pubK } Y}\}_{\text{pubK } M} \), \( ma \) is sent to the router \( M \) by an honest agent \( A \), and \( mb \) is also sent to the router \( M \) by an honest agent \( B \), then \( tr \) is observationally equivalent to swap \( ma \) \( mb \) \( tr \) in the view of the Spy.

**Lemma 11**

\[
[tr \in \text{oneOnionSession } k M; \]
\[
ma = \{\text{Nonce } n_0, \text{Agent } Y, \{\text{Nonce } n\}_{\text{pubK } Y}\}_{\text{pubK } M}; \]
Says \( A \) \( ma \) \( \in \) \set {tr}; Says \( B \) \( M \) \( mb \) \( \in \) \set {tr};
\( A \not\in \text{bad} \); \( M \not\in \text{bad} \); \( B \not\in \text{bad} \); \( n_0 \not\equiv n \vee Y \not\equiv \text{bad} \)
\[\implies \text{obsEquiv Spy } tr \ (\text{swap } ma \ mb \ tr)\]

**D. Proving anonymity properties**

Message \( ma' \) is forwarded to \( B \) by the router \( M \), and is originated by some honest agent, and the nonce \( n \) satisfies a constraint \( \text{cond } tr \ M \ n \), which will be explained in details later, then spy cannot be sure of the honest agent who originates \( ma' \). Namely, the sender anonymity holds for the intruder w.r.t. the honest agents who send messages to \( M \) in the session modelled by \( tr \).

**Lemma 12**

\[
[tr \in \text{oneOnionSession } k M; \]
\[
ma' = \{\text{Nonce } n\}_{\text{pubK } Y}; \]
Says \( M \) \( B \) \( ma' \) \( \in \) \set {tr}; regularOrig \( ma' \) \( tr \);
\( M \not\equiv \text{bad} \); \( \text{cond } tr \ M \)
\[\implies \text{senderAnonymity } (\text{senders } tr \ M - \text{bad})\]
Spy \( ma' \) \( tr \) \( \text{oneOnionSession } k M \), where senders \( tr \ M \equiv \{A;\exists m.\text{Says } A \ m \ m \in \set {tr}\}, \text{ and cond } tr \ M \equiv \forall A \ n_0 \ Y.\text{Says } A \ M \ \{\text{Nonce } n_0, \text{Agent } Y, \{\text{Nonce } n\}_{\text{pubK } Y}\}_{\text{pubK } M} \in \set {tr} \rightarrow (Y \not\equiv \text{bad} \vee n_0 \not\equiv n)\]

The premise \( \text{cond } tr \ M \ n \) says that if a nonce \( n \) is originated in a message \( \{\text{Nonce } n_0, \text{Agent } Y, \{\text{Nonce } n\}_{\text{pubK } Y}\}_{\text{pubK } M} \) in the trace \( tr \), then either \( Y \not\equiv \text{bad} \) or \( n_0 \not\equiv n \), this guarantees the secrecy of \( n \).

The last result is about the linkability of a sender \( A \) and a peeled onion \( ma \). Suppose that an honest agent \( A \) sends a message \( m \) to the router \( M \), and an agent \( B \) receives a message \( ma \) from \( M \), the intruder cannot link the message \( ma' \) with the agent \( A \) provided that there exists at least one agent \( X \) who is not \( A \) and sends a message to \( M \).

**Lemma 13**

\[
[tr \in \text{oneOnionSession } k M; \]
\[
ma' = \{\text{Nonce } n\}_{\text{pubK } Y}; \]
Says \( M \) \( B \) \( ma' \) \( \in \) \set {tr}; regularOrig \( ma' \) \( tr \);
Says \( A \) \( m \) \( \in \) \set {tr}; \( A \not\equiv \text{bad} \); \( M \not\equiv \text{bad} \);
\( \exists X.\text{Says } X \ M \ mx \in \set {tr} \wedge X \not\equiv A \wedge X \not\equiv \text{bad} \); \( \text{cond } tr \ M \ n \)
\[\implies \text{let } AS= \text{senders } tr \ M - \text{bad } \in\]
unlinkability \( AS \) \( A \ m \) (oneOnionSession \( k M \))

**VII. CONCLUSION AND FUTURE WORK**

In this work, we formalise the notion of provable anonymity in the theorem prover Isabelle/HOL. First we propose an inductive definition of message distinguishability based on the observer’s knowledge, then define the message equivalence as the negation of message distinguishability. Next, we define observational equivalence of two traces using the message equivalence, and define the semantics of anonymity properties in an epistemic logical framework. In the end, we inductively formalise the semantics of the onion routing protocols, and formally prove that sender anonymity and unlinkability hold for the protocol in Isabelle/HOL.

When we prove that properties such as sender anonymity hold for a trace under consideration, we need to consider the existence of another trace which is observationally equivalent to the given trace, but differs, for example, in the sender of some message. This is the essence of information hiding on the senders or the linkage between a message and its sender, which makes the analysis of anonymity different from analysis on secrecy and authentication. For secrecy and authentication, normally the focus is on individual traces. However, the observer decides whether two traces are observationally equivalent according to his knowledge obtained in the two traces, which usually boils down to the secrecy of some terms. Therefore, the induction proof method used in the analysis of secrecy properties can still be used here. This may be the relation between analysis on classical protocol properties on secrecy and that on anonymity properties.

In future, we will apply our framework to more case studies. We would also like to check whether our framework can be easily generalised to model different kinds of privacy and information hiding properties and to model protocols which allow more cryptographic primitives. Theoretically, we believe this inductive approach can be extended because only additional induction rules are required. In particular, it is interesting for us to find out whether the method of constructing an observationally equivalent trace using the swap function is general enough.

**REFERENCES**


