A Strand Space Approach to Provable Anonymity

Yongjian Li[1] and Jun Pang[2]

[1] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences
[2] Faculty of Science, Technology and Communication, University of Luxembourg

Abstract. We formalize in the strand space theory the notion of provable anonymity. Bundle in a strand space is used to formalize a session of a protocol. Behaviors of an observer can then be formalized as extensions of a bundle. Reinterpretation function can be naturally derived from the mapping from one message term of an edge of a bundle in a strand space to that in another strand space. We formally define observational equivalence on bundles and use it to formalise anonymity properties. The novelty of our theory lies in the observational model and the construction of reinterpretation functions in the strand space theory. We build our theory in Isabelle/HOL to achieve a mechanical framework for the analysis of anonymity protocols.

1 Introduction

Nowadays, people are getting used to carry out their daily activities through networked distributed systems, e.g., online social networks, location-based application, providing electronic services to users. In these systems, people become more and more concerned about their privacy and how their personal information have been used. Anonymity is one of the desired properties of such systems, referring to the ability of a user to own some data or take some actions without being tracked down. For example, a user wants to keep anonymous when visiting a particular website or posting a message on a public bulletin board.

Due to its subtle nature, anonymity has been the subject of many research paper. For instance, the proposed definitions aim to capture different aspects of anonymity (either possibilistic [1–5] or probabilistic [6–11]). Formal verification of anonymity has been applied to a number of application domains, including electronic voting [12, 13], electronic cash protocols [14], file sharing [15, 16] and electronic healthcare [17]. However, automatic approaches to the formal verification of anonymity have mostly focused on the model checking approach on systems with fixed configurations [1, 6, 4, 9], while theorem proving seems to be a more suitable approach when dealing with systems of infinite state spaces [18]. In this paper, we extend our previous effort on formalising provable anonymity in a powerful general-purpose theorem prover, Isabelle/HOL [19], to semi-automatically verify anonymity properties.

In the epistemic framework of provable anonymity [3], the notion of observational equivalence of traces plays an important role. Essentially, two traces are considered equivalent if an intruder cannot distinguish them. The distinguishing ability of the intruder is formalized as the ability to distinguish two messages, which is in turn based on message structures and relations between random looking messages. The notion of *reinterpretation function* is central in the provable anonymity framework – proving two traces equivalent essentially is boiled down to prove the existence of a reinterpretation function. Our formalization [20] of provable anonymity in Isabelle/HOL relies on inductive definitions of message distinguishability and observational equivalence on traces observed by the intruder. This makes our theory differ from its original proposal.

Our main contribution of this paper is twofold: a proposal of formalizing provable anonymity in the strand space theory [21–23] and its implementation in a theorem prover. We briefly discuss the novelties of our work below:

- We define an observational model of a passive intruder, meaning that the intruder does not actively modify the messages or inject new messages. The intruder only analyzes or synthesizes new messages to tell the difference between his observation on sessions. These analyzing and synthesizing actions are naturally represented by extensions of a bundle by adding separation and decryption (or concatenation and encryption) actions.
- We propose a notion of *reinterpretation mapping*, which can be naturally derived from the mapping from one message term of an edge of a bundle in a strand space to that in another strand space. Intuitively, a reinterpretation mapping requires that the relation, composed of the corresponding message pairs, should be *single_valued*. Furthermore, such a reinterpretation mapping should remain valid after applying the analyzing and synthesizing extension operations of a bundle. Combining the concepts of reinterpretation mapping with that of extensions of a bundle, we propose an (adapted) definition of observational equivalence between two sessions, which are represented by a bundle in two strand spaces. Thus in the framework, we naturally incorporate the concept of reinterpretation function which is extensively used in [3].
- We proceed to formalize anonymity properties, i.e., sender anonymity and unlinkability, in an epistemic framework as in [3]. We then define the semantics of an anonymity protocol, e.g., Onion Routing [24, 25], in the strand space theory, and formally prove that the protocol realizes sender anonymity and unlinkability.
- We build our theory in Isabelle/HOL [19] to have a mechanical framework for the analysis of anonymity protocols. We illustrate the feasibility of the mechanical framework through the case study on Onion Routing.

In this paper, we assume readers have some knowledge with Isabelle/HOL syntax and present our formalization directly without elaborated explanation. Notably, a function in Isabelle/HOL syntax is usually defined in a curried form instead of a tuple form, that is, we often use the notation $f\ x\ y$ to stand for $f(x, y)$. We also use the notation $[\![A_1; A_2; ...; A_n]\!] \Longrightarrow B$ to mean that with assumptions $A_1, \ldots, A_n$, we can derive a conclusion B.

2 Preliminaries

The basic notations and terminologies are mainly taken from [23].

2.1 Messages

The set of messages is defined using the BNF notation:

$$h ::= \mathsf{Agent}\ A \mid \mathsf{Nonce}\ N \mid \mathsf{Key}\ K \mid \mathsf{MPair}\ h_1\ h_2 \mid \mathsf{Crypt}\ K\ h$$

where A is an element from a set of agents, N from a set of nonces, and K from a set of keys. Here we use K^{-1} to denote the inverse key of K. $\mathsf{MPair}\ h_1\ h_2$ is called a composed message. $\mathsf{Crypt}\ K\ h$ represents the encryption of message h with K. We use the free encryption assumption, where $\mathsf{Crypt}\ K\ h = \mathsf{Crypt}\ K'\ h'$ if and only if $K = K'$ and $h = h'$. The set of all messages is denoted by $\mathsf{Messages}$. Terms of the form $\mathsf{Agent}\ A$, $\mathsf{Nonce}\ N$, or $\mathsf{Key}\ K$ are said to be atomic. The set of all atomic messages is denoted by Atoms. A message h is a text message if $h \neq \mathsf{Key}\ K$ for any K. The set of all atomic text messages is denoted by T.

In an asymmetric-key protocol model, an agent A has a public key $\mathsf{pubK}\ A$, which is known to all agents, and a private key $\mathsf{priK}\ A$. $\mathsf{pubK}\ A$ is the inverse key of $\mathsf{priK}\ A$ ($(\mathsf{priK}\ A)^{-1} = \mathsf{pubK}\ A$), and vice versa. In a symmetric-key model, each agent A has a symmetric key $\mathsf{shrK}\ A$. The inverse key of $\mathsf{shrK}\ A$ is itself ($(\mathsf{shrK}\ A)^{-1} = \mathsf{shrK}\ A$). We also assume that (1) asymmetric keys and symmetry keys are disjoint; (2) the functions shrK, pubK and priK are injective, e.g., if $\mathsf{shrK}\ A = \mathsf{shrK}\ A'$ then $A = A'$. The public key, private key, and shared key of an agent are long-term because the agent holds them forever. In contrast, some keys are created and used only in a session by some agents, and these keys are short-term. In the following, we abbreviate $\mathsf{Crypt}\ K\ h$ as $\{\!|h|\!\}_K$, and $\mathsf{MPair}\ h_1 \ldots \mathsf{MPair}\ h_{n-1}\ h_n$ as $\{\!|h_1, \ldots, h_{n-1}, h_n|\!\}$. Such abbreviations are supported in Isabelle by syntax translation [19]. In order to reduce the number of $\{\!|$ or $|\!\}$ for readability, we abbreviate $\mathsf{Crypt}\ K\ (\mathsf{MPair}\ h_1 \ldots \mathsf{MPair}\ h_{n-1}\ h_n)$ as $\{\!|h_1, \ldots, h_{n-1}, h_n|\!\}_K$ in this paper.

2.2 Strands and Strand Space

Actions. The set of actions that agents can take during an execution of a protocol include send and receive actions. We denote send and receive actions by a set of two signs $\mathsf{Sign} = \{+, -\}$, respectively.
Events. An event is a pair (σ, t), where $\sigma \in \mathsf{Sign}$ and $t \in \mathsf{Messages}$.
Strands and strand spaces. A protocol defines a sequence of events for each agent's role. A strand represents a sequence of an agent's actions in a particular protocol run, and is an instance of a role. A strand space is a mapping from a strand set Σ to a trace $\mathsf{SP} : \Sigma \Rightarrow (\mathsf{Sign} \times \mathsf{Messages})\ \mathsf{list}$.

- A node is a pair (s, i), with $s \in \Sigma$ and $0 \leq i < \mathsf{length}\ (\mathsf{SP}\ s)$. We use $n \in$ strand s to denote that a node $n = (s, i)$ belongs to the strand s. The set of all nodes in SP is denoted as $\mathsf{Domain}\ SP$. Namely, $\mathsf{Domain}\ SP = \{(s, i).s \in \Sigma \wedge i < \mathsf{length}\ (SP\ s)\}$.

- If $n = (s, i)$ and $(\mathsf{SP}\ s)!i = (\sigma, g)$, where $(\mathsf{SP}\ s)!i$ means the i-th element in the strand s. Then we define $\mathsf{strand}\ SP\ n$, $\mathsf{index}\ SP\ n$, $\mathsf{term}\ SP\ n$ and $\mathsf{sign}\ n$ to be the strand, index, term and sign of the node n respectively, namely $\mathsf{strand}\ SP\ n = s$, $\mathsf{index}\ SP\ n = i$, $\mathsf{term}\ n\ SP = g$ and $\mathsf{sign}\ n = \sigma$. A node is positive if it has sign $+$, and negative if it has sign $-$.
- If $n, n' \in \mathsf{Domain}\ SP$, the relation $n \Rightarrow_{SP} n'$ holds between nodes n and n' if $n = (s, i)$ and $n' = (s, i+1)$. This represents event occurring on n followed by that occurring on n'.
- If $n, n' \in \mathsf{Domain}\ SP$, the relation $n \rightarrow_{SP} n'$ holds for nodes n and n' if $\mathsf{strand}\ SP\ n \neq \mathsf{strand}\ SP\ n'$, $\mathsf{term}\ SP\ n = \mathsf{term}\ SP\ n'$, $\mathsf{sign}\ SP\ n = +$ and $\mathsf{sign}\ SP\ n' = -$. This represents that n sends a message and n' receives the message. Note that we place an additional restriction on the relation $\rightarrow$ than that in [21, 22], we require $\mathsf{strand}\ SP\ n \neq \mathsf{strand}\ SP\ n'$, i.e., n and n' are in different strands, which means that actions of sending or receiving a message can only occur between different strands.
- A term g originates in a strand space from a node $n \in \mathsf{Domain}\ SP$ iff $\mathsf{sign}\ SP\ n = +$ and $g \sqsubset \mathsf{term}\ SP\ n$, and whenever n' precedes n on the same strand, $g \not\sqsubset \mathsf{term}\ SP\ n'$. We write it $\mathsf{originate}\ SP\ g\ n$.
- A term g uniquely originates in a strand space from node n iff g originates on a unique node n. Nonces and other freshly generated terms are usually uniquely originated. We write it $\mathsf{uniqOrig}\ SP\ g\ n$.

Bundles. A bundle $b = (N_b, E_b)$ in a strand space SP is a finite subgraph of the graph $(\mathsf{Domain}\ SP, (\rightarrow_{SP} \cup \Rightarrow_{SP}))$, representing a protocol execution under some configuration. N_b is the set of nodes, and E_b is the set of the edges incident with the nodes in N_b, and the following properties hold:

- b is an acyclic finite graph;
- If the sign of a node n is $-$, and $n \in N_b$, then there is a unique positive node n' such that $n' \in N_b$, $n' \rightarrow_{SP} n$ and $(n', n) \in E_b$;
- If $n' \Rightarrow_{SP} n$ and $n \in b$, then $n' \in N_b$ and $(n', n) \in E_b$.

The set of all the bundles in a strand space SP is denoted as $\mathsf{bundles}\ SP$.

Causal precedence. Let b be a graph, we define $m \prec_b n$ for $(m, n) \in E_b^+$, and $m \preceq_b n$ for $(m, n) \in E_b^*$. $\prec_b$ and $\preceq_b$ represent causal precedence between nodes.

From the definition of a bundle b in a strand space SP, we can derive that it is a casually well-founded graph [21, 22].

Lemma 1. *For a bundle b in a strand space SP, b is casually well-founded graph, and every non-empty subset of the nodes in it has $\prec_b$-minimal members.*

2.3 Intruder model

We discuss anonymity properties based on observations of the intruder. The Dolev-Yao intruder model [26] is considered standard in the field of formal symbolic analysis of security protocols – 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, often a weaker attacker model is assumed – the intruder is *passive* in the sense that he observes all network traffic, but does not actively modify the messages or inject new messages – the intruder gets a message issued by an agent from the network, then stores it for traffic analysing, and forwards it directly to its intended destination. In the strand space model, the above behavior is typically modelled by a Tee strand. Furthermore, the copied messages are only used internally for checking observation equivalence between protocol sessions.

In the study of the anonymity, we are more interested in the observational equivalence between sessions. A session is modeled by a bundle in a strand space. Observational equivalence between two session bundles is modelled by comparing the similarity between bundles which are extended from the above two bundles by analyzing and synthesizing actions. The observational equivalence holds if a one-to-one mapping always holds between the corresponding extended bundles.

2.4 Protocol Modeling using Strands

A protocol usually contains several roles, such as initiators, responders and servers. The sequence of actions of each regular agent acting some role in a protocol session is pre-defined by the protocol and represented as a parameterized strand. Parameters usually include agent names and nonces. Informally, we denote a parameter strand acting some role by $\mathsf{role}[\textit{parameter list}]$. The strands of the legitimate agents are referred to as regular strands.

A bundle can also contain penetrator strands. We explain them in more details in the next section. We now use the Onion Routing protocol [24, 25] (see Figure 1) as an example to illustrate the modelling strategy using strands. In this figure, we abbreviate Agent A as A, Nonce N as N, and pubK A as PK_A. This figure uses the case when the threshold k of the router is 2, i.e., when the router has received two messages, then it turns into the status of forwarding messages after peeling the received messages. There are four roles in this protocol: OnionInit1, OnionInit2, OnionRouter and OnionRecv. The strands of these roles are defined below:

- OnionInit1 $SP\ s\ A\ M\ Y\ N'\ N$, if the agent acting the role is A and the trace of s in the strand space SP is $[(+, \{\!|\ N',\ Y, \{\!|\ N|\!\}_{\mathsf{pubK}\ Y}|\!\}_{\mathsf{pubK}\ M})]$.
- OnionInit2 $SP\ s\ A\ M\ N$, if the agent acting the role is A and the trace of s in the strand space SP is $[(+, \{\!|N|\!\}_{\mathsf{pubK}\ M})]$.
- OnionRouter $SP\ s\ M\ k$, if the agent acting the role is M and the trace of s in the strand space SP satisfies:
  $(\forall\, i.0 \leq\ i < k \longrightarrow ((\exists\, N'\ N\ Y.\mathsf{term}\ SP\ (s,i) = \{\!|\ N',\ Y, \{\!|\ N|\!\}_{\mathsf{pubK}\ Y}|\!\}_{\mathsf{pubK}\ M})$
  $\vee\ (\ \exists\ N.\mathsf{term}\ SP\ (s,i) = \{\!|N|\!\}_{\mathsf{pubK}\ M}))) \wedge$
  $(\ \forall i.k \leq i < \mathsf{length}\ (SP\ s) \longrightarrow (\exists\ N\ N'\ Y\ j.(0 < j < k \wedge\ \mathsf{term}\ SP\ (s,j) =$
  $\{\!|\ N',\ Y, \{\!|\ N|\!\}_{\mathsf{pubK}\ Y}|\!\}_{\mathsf{pubK}\ M} \wedge \mathsf{term}\ SP\ (s,i) = \{\!|N|\!\}_{\mathsf{pubK}\ Y}))).$
- OnionRecv $SP\ s\ Y\ N$, if the trace of s in the strand space SP is $[(-, \{\!|\ N|\!\}_{\mathsf{pubK}\ Y})]$.

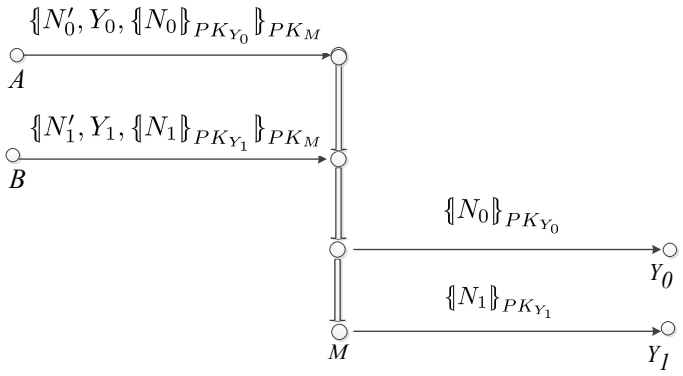

Fig. 1. Onion routing with $k = 2$.

2.5 Penetrator

The symbol bad denotes the set of all penetrators. If an agent is not in the set bad, then it is regular. The strands of the penetrators are referred to as penetrator strands. If a strand is not a penetrator one, it is referred to as a regular strand. We say a node is regular if it is at a regular strand.

There is a set of messages known to all penetrators initially, denoted as initKP, containing agent names, public keys of all agents, private keys of all penetrators, and symmetric keys initially shared between the penetrators and the server.

In the classic strand space theory, a penetrator can intercept messages and generate messages that are computable from its initial knowledge and the messages it intercepts. These actions are modeled by a set of penetrator strands, and they represent atomic deductions. More complex deduction actions can be formed by connecting several penetrator strands together.

Definition 1. *A penetrator' trace relative to* initKP *is one of the following, where* initKP *is the initial knowledge of penetrator:*

- *text message* - M a: $[(+, a)]$, *where* $a \in$ T *and* $a \in$ initKP.
- *issuing known key* - K K' : $[(+, \mathsf{Key}\ K')]$, *where* $Key\ K' \in$ initKP.
- *concatenation* - C $g\ h$: $[(-, g), (-, h), (+, \{\!|g, h|\!\})]$.
- *separation* - S $g\ h$: $[(-, \{\!|g, h|\!\}), (+, g), (+, h)]$.
- *encryption* - E $h\ K$: $[(-, \mathsf{Key}\ K), (-, h), (+, \{\!|h|\!\}_K)]$.
- *decryption* - D $h\ K$: $[(-, \mathsf{Key}\ K^{-1}), (-, \{\!|h|\!\}_K), (+, h)]$.
- *Flush* - F g: $[(-, g)]$.
- *Tee* - T g: $[(-, g), (+, g), (+, g)]$.

Roughly speaking, penetrator strands can represent two kinds of actions: analyzing messages (a combination of K and D strands, or just a separation strand); synthesizing messages (a combination of K and E strands, or just a concatenation strand). Tee strand is just for copying a message.

A bundle can be extended by adding more penetrator actions to a new bundle. The set of extended bundles of a bundle b in a strand space SP is inductively

defined in Isabelle/HOL below. Intuitively, a bundle in a strand space is a formal representation of a protocol session. If a bundle $b' \in$ extendByAnalz $SP\ b$ (or $b' \in$ extendBySynth $SP\ b$), then b' contains the same behaviors of regular agents as those in session b. However, b' contains more information which is revealed by the penetrator's analyzing (or synthesizing) actions.

In our framework, in order to check the observational equivalence between two bundles, we not only need to compare the correspondence of messages in two sessions, but also need to check the the correspondence of messages in two sessions which are extended from the original two sessions.

```
inductive_set extendByAnalz:: strand_space ⇒ graph ⇒ graph set
for SP::strand_space and b::graph where
itSelf: b ∈ bundles SP ⟹ b ∈ extendByAnalz SP b;
| Add_Decrypt: ⟦ b' ∈ extendByAnalz SP b;
Is_K_strand SP ks; (ks,0) ∉ (nodes b');
Is_D_strand SP s; (s,0) ∉ (nodes b');
(s,1) ∉ (nodes b'); (s,2) ∉ (nodes b');
(ks,0) → SP (s,0); n ∈ nodes b'; n → SP (s,1) ⟧
⟹ extendGraphByAdd1 (extendGraphByAdd2 b' ks)
s (ks,0) n ∈ extendByAnalz SP b
| Add_SepOrTee: ⟦ b' ∈ extendByAnalz SP b;
Is_Sep_strand SP s ∨ Is_Tee_strand SP s;
(node_sign SP n) = +; n ∈ (nodes b');
n → SP (s,0); (s,0) ∉ (nodes b');
(s,1) ∉ (nodes b'); (s,2) ∉ (nodes b') ⟧
⟹ extendGraphByAdd3 b' s n ∈ extendByAnalz SP b
```

```
inductive_set extendBySynth:: strand_space ⇒ graph ⇒ graph set
for SP::strand_space and b::graph where
itSelf: b ∈ bundles SP ⟹ b ∈ extendBySynth SP b
| Add_Encrypt: ⟦ b' ∈ extendBySynth SP b;
Is_K_strand SP ks; (ks,0) ∉ (nodes b');
Is_E_strand SP s; (s,0) ∉ (nodes b');
(s,1) ∉ (nodes b');(s,2) ∉ (nodes b');
(ks,0) → SP (s,0); n ∈ nodes b'; n → SP (s,1) ⟧
⟹ extendGraphByAdd1 (extendGraphByAdd2 b' ks)
s (ks,0) n ∈ extendBySynth SP b
| Add_Cat: ⟦ b' ∈ extendBySynth SP b;
Is_Cat_strand SP s; (s,0) ∉ (nodes b');
(s,1) ∉ (nodes b');(s,2) ∉ (nodes b');
n ∈ nodes b'; n → SP (s,0); n' ∈ nodes b';
n' → SP (s,1); n ≠ n'
⟧ ⟹ extendGraphByAdd1 b' s (ks,0) n ∈ extendBySynth SP b
| Add_Tee: ⟦ b' ∈ extendBySynth SP b;
Is_Tee_strand SP s; (node_sign SP n) = +;
n ∈ (nodes b'); n → SP (s,0);
(s,0) ∉ (nodes b'); (s,1) ∉ (nodes b');
(s,2) ∉ (nodes b') ⟧ ⟹ extendGraphByAdd3 b' s n ∈ extendBySynth SP b
```

3 Message Reinterpretation and Observational Equivalence on Bundles

We give a definition of message mapping from terms of a node set in a strand space to those of nodes in another strand space as follows:

```
mapping:: strand_space ⇒ strand_space ⇒ (node set) ⇒ ( msgPair set)
where mapping SP SP' NodeSet
≡ {p. ∃n. n∈ NodeSet ∧ p= (term SP n, term SP' n )}
```

Then we can naturally derive a definition from messages of a node set of a bundle in a strand space to those in another strand space. A session which is modeled by a bundle b in a strand space SP, is said to be reinterpreted to another which is modeled by b in another strand space SP', if the following conditions hold:

- Let $r =$ mapping SP SP' (nodes b), single_valued r guarantees that an agent cannot reinterpret any message differently.
- The casual relation of b in strand space SP is the same as that of b in SP'.
- For a message pair $(m, m') \in r$, if m is an atomic message, then $m = m'$. This means that an agent can uniquely identify a plain-text message he observes. An agent can only reinterpret the encrypted messages.

The corresponding formalization of Reinterp in Isabelle/HOL is given below.

```
 Reinterp::graph ⇒ strand_space ⇒ strand_space ⇒ bool where
Reinterp b SP SP' ≡
let r= mapping SP SP' (nodes b) in
single_valued r ∧
( ∀ n1 n2. (n1 → SP n2) ⟶ (n1 → SP' n2 )) ∧
( ∀ n1 n2. ( n1 ⇒ SP n2) ⟶ ( n1 ⇒ SP' n2 )) ∧
( ∀ n. n ∈ nodes b
⟶ ofAgent SP (strand n)= ofAgent SP' (strand n)) ∧
( ∀ m m'. Is_atom m ⟶ (m,m') ∈ r ⟶ m = m')
```

Next lemma says that b is also a bundle in SP' if b is a bundle in SP and Reinterp b SP SP'.

Lemma 2. $[\![$Reinterp b SP SP'; $b \in$ bundles $SP]\!] \Longrightarrow b \in$ bundles SP'

With the concepts of reinterpretation and the extensions of bundles, we can formalize the definition of observational equivalence between sessions as follows:

```
obsEquiv::graph ⇒ strand_space ⇒ strand_space ⇒ bool where
obsEquiv b SP SP' ≡
∀ b' b''. b' ∈ (extendByAnalz SP b ) ⟶
b'' ∈ (extendBySynth SP b' )⟶
( b' ∈ extendByAnalz SP' b ∧
b'' ∈ extendBySynth SP' b' ∧ Reinterp b' SP SP')
```

This definition obsEquiv means that for any extension b' of the bundle b, the reinterpretation relation will be kept between the two sessions which are modelled by b' in strand space SP and SP' respectively.

Remark 1. The intuition behind the above definition is that messages in two sessions 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. In detail,

1. For a plain-text, if the agent observes it in an action of a session, then he should observe the exact same message in the corresponding action of the other session.
2. A message looks like a random bit-string if the decryption key is not possessed by the agent. Then the corresponding message should also be like a random bit-string, which means that it is also a message encrypted by a key whose inverse key is not possessed by the observer.
3. The reinterpretation should be preserved by the synthesizing and analyzing operations on the observed messages. In the strand space theory, these operations are modelled by the penetrator strands, thus the preservation is checked by comparing the corresponding messages mapping from an extended session to another extended session which are extended by the same similar penetrator strand.

In the work of Garcia et al. [3], a reinterpretation function between two message sequences is used as a underlining concept. In our work, the single-valued requirement of the message mapping between two bundles gives a sufficient condition for the existence of a reinterpretation function. Moreover, the bundle extensions give a mechanical way to derive the reinterpretation function.

4 Anonymity Properties

Using the observational equivalence relations over a set of possible observation equivalent bundles, we can formally introduce epistemic operators [3] as follows:

```
diamond :: graph ⇒ strand_space set⇒ strand_space
⇒ assertONBundle ⇒ bool where
diamond b SPS SP Assert ≡ ∃ SP'. SP' ∈ SPS
∧ ((obsEquiv b SP SP') ∧ Assert b SP')

box :: graph ⇒ strand_space set ⇒ strand_space
⇒ assertONBundle ⇒ bool where
box b SPS SP Assert ≡
∀ SP' ∈ SPS. (obsEquiv b SP SP') ⟶ (Assert b SP')
```

Intuitively, $b \models \Box\ bs\ \varphi$ means that for any bundle b' in bs, if b' is observationally equivalent to b, then b' satisfies the assertion φ. On the other hand,

$b \models \Diamond\ trs\ \varphi$ means that there is a bundle b' in trs, b' is observationally equivalent to b and b' satisfies the assertion φ. 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.

Suppose that b is a bundle of a protocol in which a message m is originated by some agent. We say that b provides sender anonymity w.r.t. the anonymity set AS and a set of possible runs if it satisfies:

```
origInBundle::agent ⇒ msg ⇒ graph ⇒ strand_space ⇒ bool where
origInBundle A g b SP ≡
∃ n. n ∈ nodes b ∧ originate SP g n

senderAnonymity::agent set ⇒ msg ⇒ graph
⇒ strand_space set⇒ strand_space ⇒ bool where
senderAnonymity AS g b SPS SP ≡
( ∀ X. X:AS ⟶ diamond b SPS SP (origInBundle X g))
```

Here, AS is the set of agents who are under consideration, and SPS is the set of all the strand spaces where b represents a protocol session. Intuitively, this definition means that each agent in AS can originate g in a session which is represented by b in SP. Therefore, this means that B cannot be sure of anyone who originates this message in the session.

5 A Case Study: Onion Routing

Onion Routing [24, 25] provides both sender and receiver anonymity for communication over the Internet and servers as the basis of the Tor network [27]. Its main idea is based on Chaum's mix cascades [28] that messages in Onion Routing have a layered encryption (thus called *onions*) and travel from source to destination via a sequence of proxies (called *onion routers*). Each onion router can decrypt (or peel) one layer of a received message and forward the remainder to the next onion router. To disguise the relations between incoming and outgoing messages, an onion router collect incoming messages until it has received k messages, permutes the messages and sends in batch to their intended receivers.

5.1 Modeling Onion Routing

We model a simplified version of Onion Routing with only one onion router as done in [3]. We assume a set of users AS and one 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. We define its initiator and receiver and router strands. For instance, we define the two kinds of an initiator strands as follows:

```
is_initiator1::strand_space ⇒ sigma ⇒ agent ⇒ agent ⇒ nat
⇒ nat ⇒ bool where
is_initiator1 SP s M Y N0 N ≡
(SP s)=[(+, (Crypt (pubEK M) {|(Nonce N0),(Agent Y),
     Crypt (pubEK Y) (Nonce N)|}))]
∧uniqOrig (Nonce N) (s,0)
∧uniqOrig (Nonce N0) (s,0)

is_initiator2::strand_space ⇒ sigma ⇒ agent ⇒ nat ⇒ bool where
is_initiator2 SP s M N ≡
(SP s)=[(+, Crypt (pubEK M) (Nonce N) )]
∧uniqOrig (Nonce N) (s,0)
```

Next we define the strands in a strand space of onion protocol to be the union of the above kinds strands and penetrator strands.

```
onionStrandSpec:: agent ⇒ strand_space ⇒ bool where
onionStrandSpec M SP≡
∀ s. (Is_penetrator_strand SP s ∨
(∃ Y N0 N. is_initiator1 SP s M Y N0 N) ∨
(∃ N. is_initiator2 SP s M N) ∨
( ∃ k. is_router SP s M k ∧ (ofAgent SP s=M)) ∨
( ∃ Y N. is_recv SP s Y N)
onionStrandSpaces::agent ⇒ strand_space set where
onionStrandSpaces M≡={SP. onionStrandSpec M SP}
```

5.2 An overview of our proof strategy

In the following sections, we will formalize and prove the anonymity properties of Onion Routing. Due to the complexity of the epistemic operators in property definitions, the proof is rather envolved. We give an overview of our formalization and the main proof steps.

We will formalize the sender anonymity of Onion Routing in the view of a Spy for a session w.r.t. a set of honest agents and all possible equivalent bundles. Consider a session, which is modelled by a bundle b in a strand space SP, according to the definitions of epistemic operators, which are used in the definition of sender anonymity, we need to construct another strand space SP' which satisfies the following two conditions:

(1) SP' is still an Onion routing strand space.
(2) b in strand space SP is observationally equivalent to b in SP'. That is to say, obsEquiv b SP SP'. In order to show this, by the definition of obsEquiv, we need to prove that for any bundle $b' \in$ extendByAnalz SP b, $b'' \in$ extendBySynth SP' b' and Reinterp b'' SP SP'.

Whether two sessions are observationally equivalent for a protocol depends on the knowledge of the intruder after his observation of the two sessions. Therefore,

we need to discuss some secrecy upon on the intruder's knowledge. We introduce a new predicate:

$$\mathsf{nonLeakMsg}\ g\ M \equiv \forall\ B\ N_0\ N.(g = (\mathsf{Crypt}\ (\mathsf{pubK}\ M)\ \{\!|\mathsf{Nonce}\ N_0, \mathsf{Agent}\ B, \mathsf{Crypt}\ (\mathsf{pubK}\ B)(\mathsf{Nonce}\ N)|\!\})) \longrightarrow (B \notin \mathsf{bad}\ \vee\ N_0 \neq N)$$

Formally, $\mathsf{nonLeakMsg}\ m\ M$ specifies that if a message m has the form of $\mathsf{Crypt}\ (\mathsf{pubK}\ M)\ \{\!|\mathsf{Nonce}\ N_0, \mathsf{Agent}\ B, \mathsf{Crypt}\ (\mathsf{pubK}\ B)(\mathsf{Nonce}\ N)|\!\}$, then either $B \notin \mathsf{bad}$ or $N_0 \neq N$. This specifies a non-leakage condition of nonce part N_0 in a message of the aforementioned form which is sent to the router even if its nonce part N is forwarded to the intruder.

5.3 Message swapping

In this section, we present a method for the construction of an observationally equivalent session.

The swap function. We define a function $\mathsf{swapMsg}\ g\ h\ msg$, which swaps g with h if either g or h occurs in msg. Then we extend the *swap* operation naturally to events (applying to the message field of an event) and to strand space (applying to the message field of every event in a strand).

```
primrec swapMsg::msg ⇒ msg ⇒ msg ⇒ msg where
swapMsg g h (Nonce na) =
(if (g=(Nonce na)) then h else if (h=(Nonce na))
then g else (Nonce na)) |
swapMsg g h (Agent A ) =
(if (g=(Agent A)) then h else if (h=(Agent A))
then g else (Agent A)) |
swapMsg g h (Crypt K m ) =
(if (g= (Crypt K m )) then h else if (h= (Crypt K m ))
then g else (Crypt K (swapMsg g h m)))

swapSignMsg::msg ⇒ msg ⇒ (Sign × msg) ⇒ (Sign × msg) where
swapSignMsg g h sMsg ≡ (fst sMsg, swapMsg g h (snd sMsg))

definition swapStrandSpace::msg ⇒ msg ⇒ strand_space ⇒ strand_space
where swapStrandSpace g h SP ≡(%s. if ((Is_D_strand SP s)
∧ (node_term SP (s,1)=g ∨ node_term SP (s,1)=h))
then [(-,node_term SP (s,0) ),
(-,swapMsg g h (node_term SP (s,1)) ),
(+,plainTxt (swapMsg g h (node_term SP (s,1)))) ]
else if ((Is_E_strand SP s)
∧ (node_term SP (s,2)=g ∨ node_term SP (s,2)=h))
then [(-,node_term SP (s,0) ),
(-,plainTxt (swapMsg g h (node_term SP (s,2)))),
(+,swapMsg g h (node_term SP (s,2)))]
else (map (swapSignMsg g h) (SP s)))
```

Here plainTxt g is a function which returns the plain text of g which is of an encrypted form. E.g., plainTxt $\{|\mathsf{Nonce}\ N|\}_{\mathsf{pubK}\ Y} = \mathsf{Nonce}\ N$. We emphasize that g and h are two messages of encrypted form when we use the definition swapStrandSpace $g\ h\ SP$ in this work.

In strand space SP, if message $g(h)$ is uniquely originated in node $n(n')$, $g(h)$ is not a subterm of $h(g)$, $n(n')$ is in Domain SP, then $g(h)$ is uniquely originated in node $n'(n)$. Here we also assume that $g(h)$ is an encrypted message.

Lemma 3. $[\![\mathsf{uniqOrig}\ SP\ g\ n;\ \neg g \sqsubset h;\ \neg h \sqsubset g;\ \mathsf{uniqOrig}\ SP\ h\ n';\ \mathsf{ofEncryptForm}\ g;$ ofEncryptForm $h;\ n \in \mathsf{Domain}\ SP;\ n' \in \mathsf{Domain}\ SP]\!]$
$\Longrightarrow \mathsf{uniqOrig}\ (\mathsf{swapStrandSpace}\ g\ h\ SP)\ g\ n'$

swap g h SP is an Onion Strand Space. This is stated as a lemma below.

Lemma 4. $[\![SP \in \mathsf{onionStrandSpaces}\ M;\ \mathsf{term}\ SP\ (s,0) = g;\ \mathsf{term}\ SP\ (s',0) = h;\ \mathsf{is\_initiator}\ M\ SP\ s\ g;\ \mathsf{is\_initiator}\ M\ SP\ s'\ h]\!]$
$\Longrightarrow (\mathsf{swapStrandSpace}\ g\ h\ SP) \in \mathsf{onionStrandSpaces}$,
where $\mathsf{is\_initiator}\ M\ SP\ s\ g \equiv (\exists Y\ N_0\ N.\ \mathsf{is\_initiator1}\ M\ SP\ s\ Y\ N_0\ N \wedge g = \mathsf{Crypt}\ (\mathsf{pubK}\ M)\{|\mathsf{Nonce}\ N_0, \mathsf{Agent}\ B, \mathsf{Crypt}\ (\mathsf{pubK}\ B)\ (\mathsf{Nonce}\ N)|\}) \vee$
$(\exists N.\ \mathsf{is\_initiator2}\ SP\ s\ M\ N \wedge g = \mathsf{Crypt}\ (\mathsf{pubEK}\ M)\ (\mathsf{Nonce}\ N)$.

Alignment properties. Now we first define a predicate, $\mathsf{initBundle}\ SP\ b \equiv b \in \mathsf{bundles}\ SP \wedge (\forall s.(\exists i.(s,i) \in \mathsf{nodes}\ b) \longrightarrow \mathsf{is\_regular\_strand}\ SP\ s \vee \mathsf{Tee}\ SP\ s)$. We can show that the relation, $r = \mathsf{mapping}\ SP\ SP'\ \mathsf{nodes}\ b$, which is composed of the corresponding message pairs of two sessions, which are modelled by b in SP and b in swapStrandSpace $g\ h\ SP$ respectively, is single_valued. Here we also assume that initBundle $SP\ b$.

Lemma 5. $[\![b \in \mathsf{bundles}\ SP;\ SP \in \mathsf{onionStrandSpaces}\ M;\ \mathsf{term}\ SP\ (s,0) = g;$ term $SP\ (s',0) = h;$ is_initiator $M\ SP\ s\ g;$ is_initiator $M\ SP\ s'\ h;$ initBundle $SP\ b;$ $SP' = \mathsf{swapStrandSpace}\ g\ h\ SP;\ r = \mathsf{mapping}\ SP\ SP'\ (\mathsf{nodes}\ b)]\!]$
$\Longrightarrow \mathsf{Reinterp}\ b\ SP\ SP'$

After applying analyzing operations pairwise on b, we can extend b to b', let $SP' = (\mathsf{swapStrandSpace}\ g\ h\ SP)$, then we also have $b'' \in \mathsf{extendsByAnalz}\ b'SP'$ and $\mathsf{Reinterp}\ b''\ SP\ SP'$. After applying synthesizing operations pairwise on the b' in Lemma 5, we obtain another bundle b'', let $r = \mathsf{mapping}\ SP\ SP'\ \mathsf{nodes}\ b'$, if M is not in bad, and both $\mathsf{nonLeakMsg}\ g\ M$ and $\mathsf{nonLeakMsg}\ h\ M$, then we also have $b'' \in \mathsf{extendsBySynth}\ b'(\mathsf{swapStrandSpace}\ g\ h\ SP)$ and $\mathsf{Reinterp}\ b''\ SP\ SP'$.

Observational equivalence between b and swap g h b. Next we show that b in SP is observationally equivalent to b in swap $g\ h\ SP$ if the following constraints are satisfied: $g = \{|\mathsf{Nonce}\ n_0, \mathsf{Agent}\ Y, \{|\mathsf{Nonce}\ n|\}_{\mathsf{pubK}\ Y}|\}_{\mathsf{pubK}\ M}$, g is sent to the router, and h is also sent to the router M, and both g and h satisfy the nonLeakMsg conditions.

Lemma 6. $[\![SP \in \mathsf{onionStrandSpaces}\ M;\ b \in \mathsf{bundles}\ SP;\ \mathsf{initBundle}\ SP\ b;\ g = \mathsf{Crypt}\ (\mathsf{pubEK}\ M)\{\!|(\mathsf{Nonce}\ N_0), (\mathsf{Agent}\ Y), (\mathsf{Crypt}\ (\mathsf{pubEK}\ Y)\ (\mathsf{Nonce}\ N))|\!\};\ \mathsf{term}\ SP\ n = g;\ n \in \mathsf{nodes}\ b;\ \mathsf{term}\ SP\ n' = h;\ n' \in \mathsf{nodes}\ b;\ M \notin \mathsf{bad};\ \mathsf{nonLeakMsg}\ g\ M;\ \mathsf{nonLeakMsg}\ h\ M;\ \mathsf{is\_initiator}\ M\ SP\ (\mathsf{strand}\ n)\ g;\ \mathsf{is\_initiator}\ M\ SP\ (\mathsf{strand}\ n')\ h]\!] \Longrightarrow \mathsf{obsEquiv}\ b\ SP\ (\mathsf{swap}\ g\ h\ SP)$

5.4 Proving anonymity properties

Let us give two preliminary definitions: the senders in a bundle, and a predicate nonLeakBundle $b\ M$ specifying that b is a bundle where each honest agent sends a message m which satisfies nonLeakMsg $m\ b$.

```
sendersInBundle::strand_space ⇒ graph ⇒ agent set
where sendersInBundle SP b ≡
{A.∃ s. ofAgent SP s= A ∧ (s,0) ∈ nodes b
((∃ Y n0 n. is_initiator1 SP s M Y n0 n ) ∨
(∃ n. is_initiator2 SP s M n ))}

nonLeakBundle::strand_space ⇒ graph⇒ agent⇒bool
where nonLeakBundle SP b M ≡
∀ g n n'. ((n → SP n') ∧ n' ∈ nodes b ∧
ofAgent SP (strand n) ∉ bad ) ⟶ nonLeakMsg g M
```

Message g is forwarded to B by the router M, and is originated by some honest agent, and the bundle in SP satisfies nonLeakBundle $SP\ b\ M$, then the honest agent who originates g cannot be observed. Namely, the sender anonymity holds for the intruder w.r.t. the honest agents who send messages to M in the session modeled by b. This is summarized by the following theorem.

Theorem 1. $[\![SP \in \mathsf{onionStrandSpaces}\ M; b \in \mathsf{bundles}\ SP; g = \mathsf{Crypt}\ (\mathsf{pubEK}\ B)\ (\mathsf{Nonce}\ N);\ n \in \mathsf{nodes}\ b;\ \mathsf{sign}\ SP\ n = -\ ;\ \mathsf{regularOrig}\ (\mathsf{Nonce}\ N)\ b\ SP;\ \mathsf{term}\ SP\ n = g;\ \mathsf{nonLeakBundle}\ SP\ b\ M;\ M \notin \mathsf{bad}]\!] \Longrightarrow \mathsf{senderAnonymity}\ (\mathsf{sendersInBundle}\ SP\ b - \mathsf{bad})\ (\mathsf{Nonce}\ N)\ b\ (\mathsf{onionStrandSpaces}\ M)\ SP$

6 Conclusion and Future Work

We presented a strand space approach to provable anonymity and formally implemented it in the theorem prover Isabelle/HOL. In order to do this, we extended the classical strand space theory. We built the concept of a protocol session based on the notion of "bundle" in a strand space. In the classical strand space theory, secrecy and authentication are studied by focusing individual sessions. However, two protocol sessions are needed in order to decide observational equivalence according to the adversary's knowledge obtained in the two separate sessions – in our extended strand space theory, they are represented by a similar bundle in two different strand spaces. Moreover, an observer needs to compare corresponding messages to decide the equivalence of two sessions based

on his knowledge. In the strand space theory, knowledge deduction actions are represented by penetrator strands. Therefore, we proposed two kinds of bundle extensions: analyzing and synthesizing extensions, which improve the deduction ability of an observer. In the end, we proposed a natural definition on reinterpretation relation between two sessions. Essentially, the two compared sessions should have the same topological relation, and the message mapping of the two sessions should be single-valued. Combining reinterpretation relation and bundle extensions, we arrived at the key concept of observational equivalence between sessions. Based on this, we defined the semantics of anonymity properties in an epistemic framework and formally proved sender anonymity for the Onion Routing protocol. In the future, we plan to extend the whole theory to active intruders in the style of Dolev-Yao [26], and perform more case studies.

Acknowledgments. The first author, Yongjian Li, was supported by a grant 61170073 from the National Natural Science Foundation of China.

References

1. Schneider, S., Sidiropoulos, A.: CSP and anonymity. In: Proc. 4th European Symposium on Research in Computer Security. Volume 1146 of LNCS., Springer (1996) 198–218
2. Hughes, D., Shmatikov, V.: Information hiding, anonymity and privacy: A modular approach. Journal of Computer Security **12**(1) (2004) 3–36
3. Garcia, F.D., Hasuo, I., Pieters, W., van Rossum, P.: Provable anonymity. In: Proc. 3rd Workshop on Formal Methods in Security Engineering, ACM (2005) 63–72
4. Chothia, T., Orzan, S.M., Pang, J., Torabi Dashti, M.: A framework for automatically checking anonymity with μCRL. In: Proc. 2nd Symposium on Trustworthy Global Computing. Volume 4661 of LNCS., Springer (2007) 301–318
5. Arapinis, M., Chothia, T., Ritter, E., Ryan, M.D.: Analysing unlinkability and anonymity using the applied pi calculus. In: Proc. 23rd IEEE Computer Security Foundations Symposium, IEEE CS (2010) 107–121
6. Shmatikov, V.: Probabilistic model checking of an anonymity system. Journal of Computer Security **12**(3/4) (2004) 355–377
7. Halpern, J.Y., O'Neill, K.R.: Anonymity and information hiding in multiagent systems. Journal of Computer Security **13**(3) (2005) 483–514
8. Bhargava, M., Palamidessi, C.: Probabilistic anonymity. In: Proc. 16th Conference on Concurrency Theory. Volume 3653 of LNCS., Springer (2005) 171–185
9. Deng, Y., Palamidessi, C., Pang, J.: Weak probabilistic anonymity. In: Proc. 3rd Workshop on Security Issues in Concurrency. Volume 180 of ENTCS. (2007) 55–76
10. Chen, X., Pang, J.: Measuring query privacy in location-based services. In: Proc. 2nd ACM Conference on Data and Application Security and Privacy, ACM Press (2012) 49–60
11. Chen, X., Pang, J.: Protecting query privacy in location-based services. GeoInformatica (2013) To appear.
12. Delaune, S., Kremer, S., Ryan, M.D.: Verifying privacy-type properties of electronic voting protocols. Journal of Computer Security **17**(4) (2009) 435–487
13. Jonker, H.L., Mauw, S., Pang, J.: A formal framework for quantifying voter-controlled privacy. Journal of Algorithms in Cognition, Informatics and Logic **64**(2-3) (2009) 89–105

14. Luo, L., Cai, X., Pang, J., Deng, Y.: Analyzing an electronic cash protocol using applied pi-calculus. In: Proc. 5th Conference on Applied Cryptography and Network Security. Volume 4521 of LNCS., Springer (2007) 87–103
15. Yan, L., Sere, K., Zhou, X., Pang, J.: Towards an integrated architecture for peer-to-peer and ad hoc overlay network applications. In: Proc. 10th Workshop on Future Trends in Distributed Computing Systems, IEEE CS (2004) 312–318
16. Chothia, T.: Analysing the mute anonymous file-sharing system using the pi-calculus. In: Proc. 26th Conference on Formal Methods for Networked and Distributed Systems. Volume 4229 of LNCS. (2006) 115–130
17. Dong, N., Jonker, H.L., Pang, J.: Formal analysis of privacy in an eHealth protocol. In: Proc. 17th European Symposium on Research in Computer Security. Volume 7459 of LNCS., Springer (2012) 325–342
18. Kawabe, Y., Mano, K., Sakurada, H., Tsukada, Y.: Theorem-proving anonymity of infinite state systems. Information Processing Letters **101**(1) (2007) 46–51
19. Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL - A Proof Assistant for Higher-Order Logic. LNCS 2283. Springer (2002)
20. Li, Y., Pang, J.: An inductive approach to provable anonymity. In: Proc. 6th Conference on Availability, Reliability and Security, IEEE CS (2011) 454–459
21. Javier Thayer, F., Herzog, J.C., Guttman, J.D.: Strand spaces: Why is a security protocol correct? In: Proc. 19th IEEE Symposium on Security and Privacy, IEEE CS (1998) 96–109
22. Javier Thayer, F., Herzog, J.C., Guttman, J.D.: Strand spaces: Proving security protocols correct. Journal of Computer Security **7**(1) (1999) 191–230
23. Li, Y., Pang, J.: An inductive approach to strand spaces. Formal Aspects of Computing **25**(4) (2013) 465–501
24. Goldschlag, D.M., Reed, M.G., Syverson, P.F.: Hiding routing information. In: Proc. 1st Workshop on Information Hiding. Volume 1774 of LNCS., Springer (1996) 137–150
25. Syverson, P.F., Goldschlag, D.M., Reed, M.G.: Anonymous connections and onion routing. In: Proc. 18th IEEE Symposium on Security and Privacy, IEEE (1997) 44–54
26. Dolev, D., Yao, A.: On the security of public key protocols. IEEE Transactions on Information Theory **29**(12) (1983) 198–208
27. Dingledine, R., Mathewson, N., Syverson, P.F.: Tor: The second-generation onion router. In: Proc. 13th USENIX Security Symposium. (2004) 303–320
28. Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM **24**(2) (1981) 84–90