Using Strand Space Model to Verify the Privacy Properties of a Fair Anonymous Authentication Scheme

Nikolaos Petrakos, Panayiotis Kotzanikolaou, Christos Douligeris

Department of Informatics, University of Piraeus, 80 Karaoli & Dimitriou, 18534 Piraeus, Greece
{npetrakos,pkotzani,cdoulig}@unipi.gr

Abstract—The strand space model has been proposed as a formal method for verifying the security goals of cryptographic protocols. Many cryptographic protocols aim not only to provide security, but also privacy properties of the communication such as anonymity. In this paper, we apply the strand space model in order to verify the security and privacy goals of a recently developed anonymous authentication scheme. We show that the strand space model can be used to formalize privacy properties such as transaction untraceability and unlinkability, user non-frameability and anonymous credential non-transferability.

Keywords—formal methods, security protocol analysis, strand space model.

I. INTRODUCTION

A security protocol is a sequence of messages between two or more principals, in which encryption is used to provide authentication, integrity of the message, or to distribute cryptographic keys for new conversations [4]. A security protocol describes a sequence of steps the participating parties must perform. These steps mainly include the transmission of messages, participating name identifiers, cryptographic keys, random numbers, time-stamps, ciphertexts and concatenation of these components. Additionally, a security (or cryptographic) protocol aims to achieve certain security properties or goals upon its completion. These may include well-known security properties such as message integrity, i.e. ensure that the data have not been modified in an unauthorized manner, message confidentiality, i.e. information is not disclosed to unauthorized entities, entity authentication, i.e. ensure the sender’s/receiver’s identity and non-repudiation or delivery or receipt, i.e. protect against a receiver’s / sender’s false denial of having received / send a message.

The design of security protocols is a very tedious and error prone task. In recent years prominent formal approaches have been developed to verify the security properties of cryptographic protocols. Following the exceptional work of Dolev and Yao [1], who formally described the capabilities and goals of the Adversary, several formal security proof methods have been proposed. The main categories of formal security protocol verification methods include BAN Logic [9], Strand Spaces [2], [3] and Scyther [5].

In this paper we focus on the strand space model, as a formal security protocol verification method. Although the strand space model has been applied in order to verify the security properties of confidentiality, integrity and authentication, it has not been widely applied in cryptographic protocols that also provide privacy properties, such as untraceability, unlinkability, non-transferability and non-frameability. In this paper, we use the strand space model as a formal method for the verification of such security properties. As a test protocol, we use a recently proposed anonymous authentication scheme for mobile devices [11]. The rest of this paper is organized as follows. In section II we describe the strand space model, as well as the authentication scheme that we will use as a test example. In section III we provide detailed security proofs of the described privacy properties, based on strand space analysis. Finally, section IV concludes this paper.

II. RELATED WORK

Initially, we briefly describe the strand space model as well as other existing formal protocol verification methods. Then, we describe the anonymous authentication scheme that we use as a test example.

A. The Strand Space Model

Formal methods in security protocols are thanks to Dolev and Yao [1], who formally described a model that captures the behavior and the capabilities of the Adversary (i.e. the opponent of the cryptographic protocol, also known as the Intruder). They utilized the notion of an “algebraic” approach by using an algebraic set of mechanisms that provide a proof whether two particular classes of protocols are secure or not. In order to formally describe what are the properties of the Adversary, they introduce the notion of the Cascade Protocols and the Name-Stamp Protocols.

Early analysis tools mainly include the BAN logic, Model Checking and Theorem Proving. In [10], Burrows, Abadi and Needman defined a relationship about what the participating principals in a security protocol are able to believe, see and say through messages. For example, if one sees a term of the form $P$ believes $Q \stackrel{K}{\leftrightarrow} P$, $P$ sees $\{X\}_K$, $P \rightarrow T$,...
then one can replace it with the term $P$ believes $Q$ said $X$, where one can think of the $P$’s, $Q$’s, $X$’s and $K$’s as variables of the appropriate type. Their approach focus on translating a protocol into idealized messages. Nonees are transformed into arbitrary new formulates that the sender believes and in $(X)_Y$, $Y$ is assumed to be a secret and is only used as a proof of identity. Each principle “believes” all the messages that creates. BAN logic has a fixed intruder system model, which can not address conspiring agents. Man in the middle attack could not be modeled proving that Needham-Schroeder protocol is correct in BAN logic.

Model Checking [8] is another approach that essentially describes all the states that a particular system can be in and then checks whether each of the states satisfies a particular property. Several application tools have been introduced to automate the process of verification like Murϕ [14] and Casper [8].

Theorem Proving [6], [7] is a method of formalizing a system as a set of possible traces and then proves that some desirable properties of the system hold. Current analysis frameworks cope with Strand Spaces [2], [3], the spi calculus [9] and Scyther [5]. The spi calculus is designed to describe and analyze security protocols basically in the context of authentication and e-commerce, by expressing security guarantees as equivalences between spi calculus processes. Scyther is an automated tool that combines model checking with elements of theorem proving, in order to prove the correctness of a protocol for an unbounded number of sessions. The strand space model is a sequence of events that represents an execution of a protocol of a legitimate party or the sequence of actions of a penetrator [2], [3]. Strand space provides a security protocol analyzer with the ability of discovering flaws of a security protocol if it has any, or even more to prove the correctness of a protocol. In this paper we utilize strand spaces in order to firstly, verify the correctness of a recently proposed fair anonymous authentication scheme. Secondly, we use this model in order to verify the claimed privacy properties of the protocol such as untraceability and unlinkability, non-frameability and non-transferability.

B. An anonymous authentication scheme for mobile devices.

We recently use a anonymous authentication scheme introduced in [11], as a test example in order to show the applicability of the strand space model, as a formal method of proving privacy properties of a given security protocol. The scheme is build based on standard crypto primitives like zero-knowledge proofs, MACs and challenge/responses. In [11], the security and privacy properties of the scheme are proven using the standard security proofs method, based on reductions and security experiments. The security properties are reduced to the intractability of the Divisible Computational Diffie Hellman problem and on the Decisional Divisible Diffie Hellman problem. Here, we will briefly describe the credential issuing and the anonymous access protocols.

1) A fair, anonymous authentication scheme: Let $I$ denote the network operator responsible to issue, update and The scheme is based on a discrete logarithm setting. Let $p$, $q$ be two sufficiently large primes such that $q | p - 1$ and let $g$ be a generator of a multiplicative group $G$ of order $p$. Let $SK_U = u$, $PK_U = g^u$ be the long-term signature key pair of $U$ in a discrete log setting with the same generator $g$. Let $SK_I$, $SK_{SP}$ be the certified signing keys of $I$ and $SP$ respectively, using any cryptosystem. Finally, let $S$ be a secret service key, shared between $I$ and $SP$.

![Figure 1. The registration protocol](image)

Verifiable and then ZKP for each $r_i$: $\sigma_i = \text{MAC}(\sigma, g^r, V_i)$. Sign the messages $\sigma_i = \text{MAC}(\sigma, g^r, V_i)$. Store for each user $PK_i, \sigma_i, SK_i, V_i, g^{\rho_i}, V_i, \sigma_i$.

User: $SK_i = u, PK_i = g^u$

Generate $n$ credentials for $SP$

User: $\sigma_i = \text{MAC}(\sigma, g^r, V_i)$

Prove in ZK that $\epsilon$ contains $PK_i$:

User: $\epsilon = \text{MAC}(\epsilon, g^r, \epsilon)$

Sign all $e_i$ values with $SK_i$:

User: $\epsilon_i = \text{MAC}(\epsilon, g^r, \epsilon_i)$

Verify $\epsilon_i$ and store values $\epsilon_i, g^{\rho_i}, g^{\epsilon_i}, V_i, \epsilon_i$.

<table>
<thead>
<tr>
<th>User: $SK_i = u, PK_i = g^u$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate $n$ credentials for $SP$</td>
</tr>
<tr>
<td>$\rho_i, \sigma_i, g^{\rho_i}, g^{\epsilon_i}, V_i, \epsilon_i$</td>
</tr>
<tr>
<td>Prove in ZK that $\epsilon$ contains $PK_i$:</td>
</tr>
<tr>
<td>$\epsilon = \text{MAC}(\epsilon, g^r, \epsilon)$</td>
</tr>
<tr>
<td>Sign all $e_i$ values with $SK_i$:</td>
</tr>
<tr>
<td>$\epsilon_i = \text{MAC}(\epsilon, g^r, \epsilon_i)$</td>
</tr>
<tr>
<td>Verify $\epsilon_i$ and store values $\epsilon_i, g^{\rho_i}, g^{\epsilon_i}, V_i, \epsilon_i$</td>
</tr>
</tbody>
</table>

Figure 1. The registration protocol
3) User access: The registered user will anonymously access the service provider $SP$, for $n$ unlinkable transactions, by using his private key $SK_U = u$, a different anonymous credential from $\{r_i, g^{\nu_i}, V_i, h_i\}$ and the corresponding value $g^{\nu_i}, i = 1, 2, .., n$ (Fig. 2). We assume that the communication between $U$ and $SP$ is anonymous in the network or data-link layers.

For each access, $U$ sends a different credential $\{r_j, g^{\nu_j}, V_j, h_j\}$ to $SP$. The $SP$ will first verify if the credential has been already used, by checking whether it belongs to the list $UsedList$. The anonymous authentication continues by verifying (with the secret key $S$) that $h_j$ is a valid MAC. Then $U$ must prove knowledge of the correct private key $SK_U = u$ by responding to the challenges $C_1, C_2$ (used for non-frameability and non-transferability purposes respectively). The $SP$ must also prepare zero-knowledge proofs that the challenges $C_1, C_2$ indeed contain the values $r_j$ and $V_j$ respectively (this is done with $c_1, K_1, z_1$ and $c_2, K_2, z_2$). The $SP$ also signs the challenges $C_1, C_2$ along with the used credential $h_j (\sigma_{SP} = SIG_{SP}(h_j, C_1, C_2))$.

$U$ first verifies $\sigma_{SP}$, and then verifies that $C_1, C_2$ contain $r_j$ and $V_j$, by checking the zero knowledge proofs. If the verifications are successful, $U$ can safely use his/her private key $u$ and generate the responses: $R_1 = C_1^{u^{-1}}$ and $R_2 = C_2^{u^{-1}}$. Finally, the $SP$ will verify that $(R_1)\nu_1 = g^{\nu_1}$ and $(R_2)\nu_2 = g^{\nu_2}$, where $g^{\nu_i}$ is contained in the authenticated credential $h_j$. If the responses are valid, $U$ is allowed to anonymously access the requested service. For further details see [11].

III. SECURITY ANALYSIS

In this section we show the protocol correctness and analyze the security properties of the proposed protocol.

A. Preliminaries and notation

We use the strand space model [2], [3] as a formal method to prove the protocol correctness. We introduce a series of claims and show that they hold. For the rest of Section III-A we will use the standard notation of [2] for strand spaces, where: $\Sigma$ is the Strand Space; $P$ is the Penetrator Strand Space; $T$ is a set of texts (atomic messages); $C$ is a bundle (i.e. an acyclic graph of events representing a valid protocol run); $K_P$ is a set of keys known to the penetrator; $n_i$ are nodes in the strand space; $\preceq$ is the precedence relationship; $\sqsubseteq$ is the subterm relationship, and $S_{A,AS}$ is our protocol.

1) Strand Space Basic Notation: A strand is a process or a sequence of events represented as graph, with the following basic rules:

- A user may send or receive a message $m$ but not both at the same time.
- When a strand receives a message $m$ we use the $(-)$ sign and we assume that there is a unique node transmitting $m$ from which the message was received.
- We may also assume that if two different strands send the same message $m$ the receipt only comes from one of them.
- When a sender transmits a message $m$ we use the $(+)$ sign and we assume that many strands may receive the same message.

Figure 2. The user access protocol.
Send and receive events on strands called nodes or vertices. Let \( M \) be the set of all possible messages. The elements of \( M \) are the terms of the messages. If \( \alpha \) is a possible message in \( M \) (\( \alpha \in M \)), then \( t \) is term in the message \( \alpha \) (\( t \in \alpha \)). There is also the subterm relation (\( \subset \)) on the terms, written \( t_0 \subset t_1 \). For example, \( N_\alpha \) is a subterm of the set \( \{ N_\alpha, A \} \). \( N_\alpha \subset \{ N_\alpha, A \} \). Terms in a message are freely generated from keys, names, texts, nonces using concatenation. In a higher level of abstraction, a strand for a legitimate user represents the actions of that user for a particular run of the protocol.

Let \( T \subset M \) be the set of texts that a user wishes to include as part of the message. Let \( K \subset M \) be the sets of keys. It holds that \( T \cap K = \emptyset \).

In \( M \) we may use the following functions: \( \text{join} : M \times M \rightarrow M \). The function \( \text{join} \) shows how elements of \( M \) can be made up of pairs of other elements of \( M \). \( \text{encr} : K \times M \rightarrow M \). The function \( \text{encr} \) shows how to take an arbitrary element of \( M \) and apply encryption with a key. \( \text{inv} : K \rightarrow K \). The inverse and encryption functions together must satisfy \( \forall m \in M, \forall k \in K \):

\[ m = \text{encr}(\text{inv}(k), \text{encr}(k, m)) = \text{encr}(k, \text{encr}(\text{inv}(k), m)). \]

Thus, associativity holds for the \( \text{join} \) function. We also include some freeness assumptions:

Axiom 1: \( \forall m, m' \in M \) and \( K, K' \in K \),

\[ \{ m \} _K = \{ m' \} _{K'} \Rightarrow m = m' \land K = K'. \]

Axiom 2: \( \forall m_0, m'_0, m_1, m'_1 \in M \) and \( K, K' \in K \):

1) \( m_0 m_1 = m'_0 m'_1 \Rightarrow m_0 = m'_0 \land m_1 = m'_1 \).
2) \( m_0 m_1 \notin \{ m'_1 \} _{K'} \), (i.e. a pair cannot be interpreted as an encryption).
3) \( \{ m_0 \} _K \notin \{ K \cup T \} \), (i.e. a pair is not a key or a base text element).
4) \( \{ m_0 \} _K \notin \{ K \cup T \} \), (i.e. an encryption is not a text or a key).

Let the function \( \text{width} : A \rightarrow N \) to be the number of terms in a term \( m \in M \). It derives that if \( m \in K \cup T \) or if \( m = \{ m_0 \} _K \) for some \( m_0 \in M \) and \( K \in K \) then \( \text{width}(m) = 1 \). If, for some \( m_0, m_1 \in A, m = m_0 m_1 \) then \( \text{width}(m) = \text{width}(m_0) + \text{width}(m_1) \).

2) Penetrator Strands: The Penetrator strand mainly includes the following actions/operations:

M. (Text message) He can send out an arbitrary text message \(< t > \) where \( t \in T \).

F. (Flushing) He can receive a message and just do nothing \(< -g > \).

T. (Tee) He can receive anything and send out two copies of it \(< -g, +g, +g > \).

C. (Concatenation) He can receive a pair of messages and can tied them together \(< -g, -h, +gh > \).

S. (Separation into components) He can receive a message which is a pair and send it away \(< -gh, +g, +h > \).

K. (Key) He can send out a key, where the key is one of the keys that he knows, like his own key \(< +K > \) where \( K \in K_P \).

E. (Encryption) If he has a key and a message he can encrypt the message with the key \(< -K, -h, +\{ h \} _K > \).

D. (Decryption) If he has a decryption key and he is having a message encrypted with that key , then he can get the message back \(< -K^{-1}, -\{ h \} _K, +h > \).

B. Protocol Correctness

The protocol of [11] satisfies the required property of agreement, originality, uniqueness and secrecy between the user \( U \), the Issuer \( I \) and the Service Provider \( SP \). If \( U \) has the proper credentials, he may access \( SP \). A strand space of the protocol is depicted in Fig. 3. On top of Figure 3 the registration phase of the protocol is depicted with the messages exchanged between the User and the Issuer, whereas on the bottom of the same figure the messages exchanged between the User and the Service Provider, once the former has received the credentials from the Issuer. We use the same notation as it concerns the terms inside the messages exchanged between the participants, as in previous section.
Definition 3.1: An infiltrated strand space \((\Sigma, \mathcal{P})\) is \(S_{\text{AAS}}\), if \(\Sigma\) is the union of four types of strands:

1) Penetrator Strands \(p \in \mathcal{P}\).
2) Initiator Strands represented by the user \(u \in \mathcal{U}\) \(\{\sigma_U, r_i, M_i, V_i, r_i, h'_i, C_1, K_1, z_1, C_2, K_2, z_2, \sigma_{SP}, g^\rho, R_1, R_2\}\) with trace:
\(\langle +\{g^\rho, \sigma_U, r_i, M_i, V_i\}PK_1,\ 
-\{\sigma_I, h'_i, V_i\}PK_1,\ 
+\{r_i, p^\rho, V_i, h'_i\}PK_{SP},\ 
-\{C_1, K_1, z_1, C_2, K_2, z_2, \sigma_{SP}\}SK_{SP},\ 
+\{g^\rho, R_1, R_2\}PK_{SP} \rangle\)
3) Issuer Strands represented by the Issuer \(d \in \mathcal{U}\) \(\{\sigma_U, r_i, M_i, V_i, \sigma_I, h'_i\}\) with trace:
\(\langle -\{g^\rho, \sigma_U, r_i, M_i, V_i\}PK_1,\ 
+\{\sigma_I, h'_i, g^\rho, V_i\}PK_1 \rangle\)
4) Responder Strands represented by the Service Provider \(s \in \mathcal{SP}\) \(\{r_i, g^\rho, V_i, h'_i, \sigma_{SP}, C_1, K_1, z_1, C_2, K_2, z_2, g^\rho, R_1, R_2\}\) with trace:
\(\langle -\{r_i, g^\rho, V_i, h'_i\},\ 
+\{\sigma_{SP}, C_1, K_1, z_1, C_2, K_2, z_2\},\ 
-\{g^\rho, R_1, R_2\} \rangle\)

where, \(\sigma_U, r_i, M_i, V_i, \sigma_I, h'_i, C_1, K_1, z_1, C_2, K_2, z_2, \sigma_{SP}, g^\rho, R_1, R_2 \in \mathcal{T}_\text{data}, (i = 0, 1, \ldots, n)\), \(PK_m = g^\rho, PK_{SP}, S \in K_{\text{keys}}\).

1) Secrecy Analysis:

Proposition 3.1 (Secrecy analysis): Suppose \(\Sigma\) is a \(S_{\text{AAS}}\) space, \(\mathcal{C}\) is a bundle of \(\Sigma\), \(\{SK_{U}, SK_{SP}, SK_{SI}, S\} \notin K_{\mathcal{P}}\) and \(h'_i\) is uniquely originating in \(\Sigma\). Let set \(\mathcal{E}\) be the set containing the values that cannot be disclosed, \(\mathcal{E} = \{SK_U, SK_{SI}, SK_{SP}, S, h'_i, \sigma_I, \sigma_U\}\) and set \(\overline{\mathcal{K}}\) be the set containing all keys whose inverse the penetrator may reveal before obtaining any value in \(\mathcal{S}\), then for every regular node \(m \in \mathcal{C}\), term \(m \in \mathcal{T}_\mathcal{P}[\mathcal{E}]\).

Proof: If there is a regular node \(m \in \mathcal{C}\) let term \(m \in \mathcal{T}_\mathcal{P}[\mathcal{E}]\), then at least one of the secrets \(\{SK_U, SK_{SP}, SK_{SI}, S, h'_i, \sigma_I, \sigma_U\}\) should be a subterm of term \(m\). In accordance to Axiom 2, secret keys cannot be subterms in any terms of the legal participants, therefore \(SK_U\) and \(SK_{SP}\) are confidential. We are interested in the secrecy of the credentials \(h'_i\) they should be subterm of term \(m\). If \(m\) is a positive regular node, then \(h'_i \subseteq \text{term}(m)\). Since \(h'_i\) is uniquely originating in \(\Sigma\), then the term \(\langle m \rangle = \{r_i, g^\rho, V_i, h'_i\}PK_{SP}\), as it concerns the access phase, it belongs to \(\mathcal{T}_\mathcal{P}[\mathcal{E}]\). Since \(SK_{U}, SK_{SP}\) are confidential this contradicts the hypothesis. Therefore, \(h'_i\) are confidential. With a similar approach we can prove the secrecy of credentials during the access phase.

2) Authentication Analysis:

Proposition 3.2 (Authentication analysis): Suppose \(\Sigma\) is a \(S_{\text{AAS}}\) space, \(\mathcal{C}\) is a bundle of \(\Sigma\) and \(\{h'_i, \sigma_U\}\) uniquely originating in \(\Sigma\), whereas \(\{SK_U, SK_{SI}, SK_{SP}, S\} \notin K_{\mathcal{P}}\). If there is a regular source strand during registration phase \(u \in \mathcal{U}\) \(\{\sigma_U, r_i, M_i, V_i, \sigma_I, h'_i, C_1, K_1, z_1, C_2, K_2, z_2, \sigma_{SP}, g^\rho, R_1, R_2\}\) of height \(C(u) = 2\) then there is a regular destination node \(d \in \mathcal{U}\) \(\{\sigma_U, r_i, M_i, V_i, \sigma_I, h'_i, C_1, K_1, z_1, C_2, K_2, z_2, \sigma_{SP}, g^\rho, R_1, R_2\}\) of height \(C(d) = 2\).

Proof: The first node on strand during registration phase is node \(n_0\) (Fig. 3) having height 1. Since \(n_0\) is a positive regular node and from the term \(\{g^\rho, \sigma_U, r_i, M_i, V_i\}PK_1\), \(\sigma_U\) is uniquely originated in \(n_0\), then there exists a negative regular node \(n_1\) to receive \(\sigma_U\). Let \(n_1\) be negative with node height \(C(d) = 1\). However, \(n_1 \Rightarrow n_2\) and from term \(n_3\) are equivalence transformation for the negative term received in \(n_3\) which is of height 2.

C. Privacy Analysis

Now that we have shown protocol correctness, we will focus on the privacy properties of the protocol, namely transaction untraceability and unlinkability, user non-frameability and transaction non-transferability.

1) Proof of untraceability and unlinkability: We will first prove the basic anonymity properties of transaction untraceability and unlinkability of users against service providers.

The only parameter that can be used by the Service Provider in order to trace or link transactions of a user is the public key \(g^\rho\) of the user. Our goal is to show that the service provider cannot use the received messages in order to reveal the users‘ public key. We will prove that by showing that the term \(g^\rho\) is uniquely originated at node \(n_0\) which lays on the User’s strand.

Proposition 3.3 (Untraceability and Unlinkability): The user public key \(g^\rho\) originates at \(n_0\) and never appears on the Service Provider strand.

Proof: \(g^\rho \not\subset \{g^\rho, \sigma_U, r_i, M_i, V_i\}PK_1\) and the sign of \(g^\rho, \sigma_U, r_i, M_i, V_i\) is positive.

However, there is no other node preceding \(n_0\) that resides on the same strand as the User’s one. It is also clear that term \(g^\rho\) never appears on the Service Provider Strand.

2) Proof of Non-Frameability: We will now prove the non-frameability property. That means that \(\mathcal{P}\) and \(\mathcal{SP}\) cannot collaborate in an efficient manner and trap user \(U\). Our goal is to prove that once \(U\) has examined in ZKP that \(C_1\) is well formed, now he can safely respond to that specific Challenge. This requires the use of the long-term secret key \(SK_U = u\) and the value \(g^\rho\) that has just been revealed by \(U\). We will prove that by showing that terms \(g^\rho, R_1\) are uniquely originated on \(U’s\) strand and \(C_1\) is uniquely originated on \(\mathcal{SP}s\) strand.

Proposition 3.4 (Non-Frameability): The anonymous authentication scheme achieves non-frameability when \(g^\rho, R_1\) uniquely originating at \(n_0, g^\rho\) never appears on the Issuer’s strand whereas term is \(C_1\) uniquely originating at \(n_0\).

Proof: \(g^\rho, R_1 \not\subset \{g^\rho, R_1, R_2\}\) and the sign of \(g^\rho, R_1, R_2\) is positive. Thus, we need to check whether \(g^\rho, R_1\) \(\not\subseteq\ n’ \), where \(n’\) is the node \(n_7\).
preceeding $n_8$ on the same strand. Since
\[ term(n') = \{ \sigma_{SP}, C_1, K_1, z_1, C_2, K_2, z_2 \} \]
and there is not any occurrence of $g^{\rho_i}, R_2$ in that specific node on the User’s strand, then
\[ g^{\rho_i}, R_1 \text{ uniquely originates on at } n_6. \]
At the same time $C_1$ uniquely originating at $n_6$ since $C_1$ does not appear in any of the above nodes.

3) Proof of Non-Transferability: We will prove at last the non-transferability property. That means that $U$ cannot transfer one or more credentials to a non-registered user. Our goal is to prove that $U$ has to use his private key $u$ in order to generate Response $R_2$. We will prove that by showing that terms $g^{\rho_i}, R_2$ are uniquely originated on $U$’s strand and $C_2$ is uniquely originated on $U$’s strand.

**Proposition 3.5 (Non-Transferability):** The anonymous authentication scheme achieves non-transferability when
\[ g^{\rho_i}, R_2 \text{ uniquely originating at } n_8, \]
and term $C_2$ uniquely originating at $n_6$. $g^{\rho_i}$ never appears on Issuer’s strand whereas term $C_2$ uniquely originating at $n_6$. $g^{\rho_i}$, $R_2$ and the sign of $g^{\rho_i}, R_1, R_2$ is positive respectively. Thus, we need to check whether $g^{\rho_i}, R_1 \not\subseteq n'$.

**Proof:** Similarly with the previous proposition, $g^{\rho_i}, R_2 \subset g^{\rho_i}, R_1, R_2$ and the sign of $g^{\rho_i}, R_1, R_2$ is positive respectively. Thus, we need to check whether $g^{\rho_i}, R_1 \not\subseteq n'$, where $n'$ is the node $n_7$, preceding $n_8$ on the same strand. $g^{\rho_i}, R_2$ uniquely originating at $n_8$ since $C_1$ does not appear in any of the above nodes.

IV. Conclusion

In this paper we use the strand space model as a formal security method, in order to verify the claimed privacy properties of a recently proposed anonymous authentication protocol for Location Based Services [11]. Our results show that the strand space model can be applied in order to model and verify privacy properties of cryptographic schemes. Our future work will include the modeling of these protocols and testing for possible security flaws using automated security analysis tools for strand spaces (e.g. [13]).

**REFERENCES**


