2011-05-27

Proofs of Correctness for Three Decentralized Authentication Protocols Using Strand Spaces

Pavan Kumar Vankamamidi
Brigham Young University - Provo

Proofs of Correctness for Three Decentralized Authentication Protocols Using Strand Spaces

Pavan Kumar Vankamamidi

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Kent E. Seamons, Chair
Jay McCarthy
Mike Jones

Department of Computer Science

Brigham Young University

June 2011

ABSTRACT

Proofs of Correctness for Three Decentralized Authentication Protocols
Using Strand Spaces

Pavan Kumar Vankamamidi
Department of Computer Science, BYU
Master of Science

Security is a major concern is today's online world. As online activities become increasingly sensitive, service providers rely on security protocols to ensure confidentiality, integrity and authentication of their users and data. Greater assurance is provided when these protocols are verified to be correct.

Strand Spaces is a method to formally analyze security protocols. The arguments are based on the messages being transmitted and received while assuming that the underlying cryptographic primitives are secure. This thesis demonstrates that the protocols Luau, pwdArmor and Kiwi are secure using Strand Spaces methodology.

Keywords: Formal proofs, pwdArmor, Kiwi, Luau, Strand Spaces

ACKNOWLEDGMENTS

I am extremely grateful to my family who put me ahead in their priorities and supported me throughout. I would also like to thank Dr. Seamons whose support, advice and kind words were of great help during my studies here at BYU.

Contents

List of Figures

Chapter 1

Introduction

A security protocol is a series of messages that are exchanged between two or more participants in order to provide key establishment, authentication and confidentiality. Due to the distributed nature of the Internet and the increasing number of applications that deal with sensitive data, security protocols are being deployed to ensure the above properties. Examples of secure protocols include SSL/TLS, PGP, S/MIME and Kerberos.

Although there exist a number of well-established and tested protocols, the wide range of application requirements constantly drives the development of new protocols. For example, S/MIME is a protocol that allows users to send and receive signed and encrypted email. It uses the same digital certificate technology (X.509) used by TLS, but has not gained much popularity because of usability issues caused by client certificate management. Hence new security protocols are constantly under development to meet specific requirements of new applications.

One of the major issues with designing new protocols is that they can be notoriously difficult to test for correctness in the face of an adversary that is very powerful and opportunistic. The Dolev-Yao model defines the adversary as an entity that can read or modify any message as it is being transmitted, inject messages of its own, delay or replay any of the messages and also initiate an instance of the protocol with a legitimate participant. It is not possible to cover all possible attack scenarios with an ad-hoc analysis, hence we need formal analysis of security protocols to verify their correctness.

One of the earliest authentication protocols was the Needham Schroeder protocol [7]. It involves the exchange of 3 messages as shown in Figure 1.1. The keys K_A and K_B are the public keys of Alice and Bob respectively. The purpose of this protocol is mutual authentication between Alice and Bob. A man-in-the-middle attack on this protocol was discovered 17 years later by Lowe [5] where the attacker can impersonate Alice to Bob. This illustrates why we need to formally analyze security protocols.

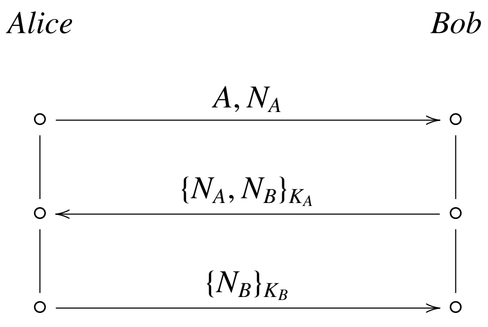


Figure 1.1: Needham Schroeder Mutual Authentication Protocol using Public Keys

Luau, pwdArmor and Kiwi are three protocols that have been developed in the Internet Security Research Lab for convenient decentralized authentication over the Internet. They have been subjected to ad-hoc threat analysis but lack a formal proof of correctness. This thesis provides a formal proof of correctness for each security protocol.

The thesis is organized as follows. Chapter 2 gives an overview of the various methods of security protocol analysis. Chapter 3 has a brief explanation and also definitions and theorems of the Strand Spaces methodology. We shall be using these to provide proofs for the three protocols in chapters 4-6. Each of these protocol chapters is organized as follows. The first section introduces the protocol and its goals. The second section provides further details and explains terms and the messages that are exchanged. The third section contains the formal analysis of the protocol. Finally, we summarize our contributions in Chapter 7 and talk about future work.

Chapter 2

Related Work

Testing security protocols for vulnerabilities is extremely important. Although a regular participant's actions are limited, an attacker is much more powerful and can participate in a protocol and manipulate its messages in various ways. Analyzing the protocol helps the designers to figure out if the protocol is secure against attackers, and if not, how the attacker can gain unauthorized access to information such as session keys. There are two high level approaches to analyzing security protocols, ad-hoc analysis and formal analysis.

2.1 Ad-hoc Analysis

Ad-hoc analysis, or threat analysis is a quick way for checking common vulnerabilities but it cannot provide a formal guarantee of protocol correctness since such analysis is not exhaustive. Many types of attacks are hard to catch during an ad-hoc analysis. It relies on experience and it is very useful during a protocol design to avoid vulnerabilities.

2.2 Formal Methods

Formal analysis provides a systematic and comprehensive method to analyze a security protocol. It thereby provides greater assurance of a protocol's correctness and allows protocol designers to find vulnerabilities that could be missed by an ad hoc analysis.

At a high level, formal methods can be categorized as model checking or formal proofs. Most of the techniques have been based on or influenced by the Dolev-Yao model [2], one of the first attempts to formalize the actions and capabilities of an adversary and also treat cryptographic operations as a black box.

2.2.1 Model Checking

Model checking methods explore the different possible states that a protocol can be in along with the penetrator's possible actions on the protocol, and check if any of the states represents an attack on the protocol. As the complexity increases so does the number of possible states, but a number of techniques are employed to counter this state-space explosion problem.

Examples of popular model checking techniques are NRL Analyzer [6] and CSP/FDR [8]. Lowe [7] had used FDR to find an attack on the Needham-Schroeder public key protocol.

2.2.2 Formal Proofs

In contrast to model checking methods, proof based methods attempt to show the correctness of a protocol rather than finding an attack on it. A typical proof includes a formal definition of the protocol as a high level model, a formal definition of the capabilities of the penetrator and the protocol's security properties, and proofs of correctness for the protocol. Two popular methods are BAN Logic [1] and Strand Spaces [3].

Chapter 3

Strand Spaces

The Strand Spaces model [3] is a framework for the formal specification and analysis of security protocols. A strand is the sequence of events that occur at that particular participant during one complete instance of a protocol. A strand space is composed of the set of strands of all the regular participants in a protocol along with any number of penetrator strands. Penetrator strands are a formalization of the Dolev-Yao [2] capabilities of the penetrator.

We make use of a set of results called authentication tests to prove correctness of the protocols by showing that the regular participants are in communication with each other and not with the penetrator strands. In the following sections, we present definitions for some of the core concepts behind Strand Spaces as specified in [3].

3.1 Definitions and Theorems

Definition 1. **A** *is the set of messages that can be sent between principals. We call elements of* **A** terms. ***A*** *is freely generated from two disjoint sets,* ***T*** *(representing texts such as nonces or names) and* ***K*** *(representing keys) by means of concatenation and encryption. The concatenation of terms g and h is denoted $g \circ h$ and the encryption of h using key K is denoted $\{h\}_K$.*

Definition 2. *A term t is a* subterm *of another term t', written $t \sqsubset t'$, if starting with t we can reach t' by repeatedly concatenating with arbitrary terms and encrypting with arbitrary keys.*

Definition 3. *A* strand *is a sequence of message transmissions and receptions, where transmission of a term t is represented as $+t$ and reception of a term t is represented as $-t$.*

Definition 4. *A strand element is called a* node.

Definition 5. *If s is a strand,* $\langle s, i\rangle$ *is the* i^{th} *node on s.*

Definition 6. *The relation* $n \Rightarrow n'$ *holds between nodes n and* n' *if* $n = \langle s, i\rangle$ *and* $n' = \langle s, i+1\rangle$.

Definition 7. *A* strand space Σ *is a set of strands.*

Definition 8. *A* regular strand *is a strand representing a legitimate participant in a protocol.*

Definition 9. *A* regular node *is a node of a regular strand.*

Definition 10. *A* penetrator strand *is a non-regular strand, i.e. an attacker.*

Definition 11. *Starting with a base case, we can inductively define the set of keys that are accessible to a penetrator and the set of safe keys.* P_0 *is the set of keys known to the penetrator initially, apart from any protocol activity. These could be the penetrator's private keys, the public keys of all the participants, etc.*

Definition 12. P *is a conservative approximation of the set of keys the penetrator may learn.* $\mathsf{P} = \bigcup_i \mathsf{P}_i$ *where* $\mathsf{P}_{i+1} = \mathsf{P}_i \cup Y$ *where* $K \in Y$ *if and only if there exists a positive regular node* $n \in \Sigma$ *and a term t such that t is a new component of n and* $K \sqsubset_{P_i^{-1}} t$.

Definition 13. *Similarly, we define safe keys by building up on the base case of* S_0 *being the set of keys K such that* $K \notin K_P$. *Clearly, the set of penetrable keys and safe keys are disjoint.*

Definition 14. *A* penetrator trace *is one of:*

M. *Text message:* $\langle +t\rangle$ *where* $t \in \boldsymbol{T}$.

C. *Concatenation:* $\langle -g, -h, +g \circ h\rangle$

S. *Separation:* $\langle -g \circ h, +g, +h\rangle$

K. *Key:* $\langle +K\rangle$ *where* $K \in \mathsf{P}$.

E. *Encryption:* $\langle -K, -h, +\{h\}_K\rangle$.

D. *Decryption:* $\langle -K^{-1}, -\{h\}_K, +h\rangle$.

These describe the capabilities of the penetrator. A penetrator is any entity on the Internet which could potentially: Read any message, modify any message as it is being transmitted, inject *messages of their own, delay and reply to any of the messages. They can also initiate an instance of the protocol with any of the participants involved.*

Definition 15. *A term t_0 is a* component *of t if $t_0 \sqsubset t$, t_0 is not a concatenated term, and every $t_1 \neq t_0$ such that $t_0 \sqsubset t_1 \sqsubset t$ is a concatenated term. Components are either atomic values or encryptions.*

Definition 16. *A component t is* new *at $n = \langle s, i\rangle$ if t is a component of $term(n)$ but t is not a component of node $\langle s,i\rangle$ for every $j < i$.*

Definition 17. *The edge $n \Rightarrow n'$ is a* transformed edge *for $a \in \mathbf{A}$ if n is positive and n' is negative, $a \sqsubset term(n)$ and there is a new component t' of n' such that $a \sqsubset t'$.*

Definition 18. *The edge $n \Rightarrow n'$ is a* transforming edge *for $a \in \mathbf{A}$ if n is negative and n' is positive, $a \sqsubset term(n)$ and there is a new component t' of n' such that $a \sqsubset t'$.*

Definition 19. *t is a* regular component *if there is a regular node n such that t is a component of $term(n)$.*

Definition 20. *$t = \{h\}_K$ is a test component for a in n if $a \sqsubset t$, t is a component of n; t is not a proper subterm of a component of any regular node $n' \in \Sigma$.*

Definition 21. *A term t* originates *at a node $n = \langle s, i\rangle$ if the sign of n is positive, $t \sqsubset term(n)$, and $t \not\sqsubset term(\langle s, j\rangle)$ for every $j < i$.*

Definition 22. *If a value originates on only one node in the strand space, we call it* uniquely originating.

Definition 23. *The edge $n \Rightarrow n'$ is a* test *for a if a uniquely originates at n and $n \Rightarrow n'$ is a transformed edge for a.*

Definition 24. *The edge $n \Rightarrow n'$ is an* outgoing test *for a in $t = \{h\}_K$ if it is a test for a in which $K^{-1} \notin \mathsf{P}$; a does not occur in any component of n other than t; and t is a test component for a in n.*

3.1.1 Authentication Tests

Authentication tests are results that have been derived using the Strand Spaces model. In order to prove authentication in a protocol, we simply need to check if the messages exchanged between the participants satisfy the constraints of either of the authentication tests. An authentication test is basically a formalization of challenge-response handshake that is used in protocols. There are three authentication tests:

1. **Authentication Test 1:** If a participant sends out a uniquely originating value such as a nonce in an encrypted form and challenges the recipient to decrypt and return the value in another form, this constitutes an outgoing test.
2. **Authentication Test 2:** If a participant sends out a uniquely originating value and receives it back in encrypted form and the challenge is to generate that encrypted form, then this constitutes an incoming test.
3. **Authentication Test 3:** If a participant receives a message in a form that can only be generated by another honest participant without explicitly requesting for it, then this constitutes and unsolicited test. This test is generally used by key servers to authenticate clients.

We now provide the formal theorems for outgoing tests and unsolicited tests, which will be used in the formal proofs for Luau, pwdArmor and Kiwi.

Definition 25. *The edge $n \Rightarrow n'$ is an* outgoing test *for a in $t = \{h\}_K$ if it is a test for a in which $K^{-1} \notin \mathsf{P}$; a does not occur in any component of n other than t; and t is a test component for a in n.*

Theorem 1. *(Authentication Test 1) If $n \Rightarrow n'$ is an outgoing test for a in t, then there exist regular nodes m and m' such that t is a component of m and $m \Rightarrow m'$ is a transforming edge for a.*

If a occurs only in component $t_1 = \{h_1\}_{K_1}$ of m', t_1 is not a proper subterm of any regular component, and ${K_1}^{-1} \notin \mathsf{P}$, then there is a negative regular node m'' with t_1 as a component.

Definition 26. *A negative node n is an* unsolicited test *for $t = \{h\}_K$ if t is a test component for any a in n and $K \notin \mathsf{P}$.*

Theorem 2. *(Authentication Test 3) If n is an unsolicited test for $t = \{h\}_K$, then there exists a positive regular node m such that t is a component of m.*

3.1.2 Example

Shown below is the Needham-Schroeder-Lowe protocol [5] in Strand Spaces format. This is the fixed version of the protocol shown in Figure 1.1.

1. Alice's strands in A[A, B, N_a, N_b] with trace:

   $$\langle +\{N_a\ A\}_{K_b},\ -\{N_a\ N_b\ B\}_{K_a},\ +\{N_b\}_{K_b} \rangle$$

2. Bob's strand in B[A, B, N_a, N_b] with trace:

   $$\langle -\{N_a\ A\}_{K_b},\ +\{N_a\ N_b\ B\}_{K_a},\ -\{N_b\}_{K_b} \rangle$$

3.2 Proof Methodology

Modelling We start by listing the assumptions such as a pre-shared secret keys between the participants. Then the given protocol needs to be specified in the Strand Spaces format. This step is also the most important. A protocol specification could be too complicated to be expressed directly in Strand Spaces format, hence, it needs to be abstracted. Strand Spaces has very limited representations of primitive operations such as encryption and decryption. Let us look at two examples of such abstraction.

One scenario is where we need to represent a Message Authentication Code (MAC). A MAC is simply a function of the message that needs to be authenticated and a shared secret key whose output is a code that can only be generated by the participants who have access to the secret key. This provides both message integrity and authentication. There is no Strand Spaces representation for a MAC, but there is for Digital Signature. In a Digital signature is a function of the message and the private key of the creator of the message. This message can now be authenticated using the corresponding public key by using the inverse of the function. We argue that MAC is equivalent to a Digital Signature without the existence of the corresponding public key. Hence, only the participants who share the private key can verify the authenticity and integrity of the message.

Another example is the representation of a TLS connection. We represent the TLS connection as a shared secret key $K_{tls(A,B)}$ between the participants A and B. If a message is transmitted from A to B within the TLS connection, we represent it as being encrypted by the TLS key $\{message\}_{K_{tls(A,B)}}$. A compromised TLS connection is represented by the penetrator having access to the key $K_{tls(A,B)}$ and subsequently to the encrypted message itself.

Authentication Tests In this step, we first identify the test components and verify that they satisfy the required constraints. We then check if these test components satisfy either of the three authentication tests defined in the earlier section. This constitutes the proof of authentication for the participants involved.

Key Secrecy For key establishment protocols, we also need to show that the final established key belongs to the set of safe keys and does not belong to the set of penetrator keys.

Chapter 4

Luau

4.1 Introduction

Most services on the Internet require users to register a new account and create a password associated with it for the purpose of authenticating the user. Remembering all these passwords is a difficult task and leads the user to adopt insecure password management techniques such as using simple passwords or reusing the same passwords across all services. Service providers have been using email as a means to authenticate users during the process of account creation and password recovery. Luau [10] is a password-based decentralized authentication protocol that leverages this trust that the service providers (or relying parties) place in email service providers (as third-party identity providers) to offload primary authentication to these identity providers and removes the need for service-specific passwords.

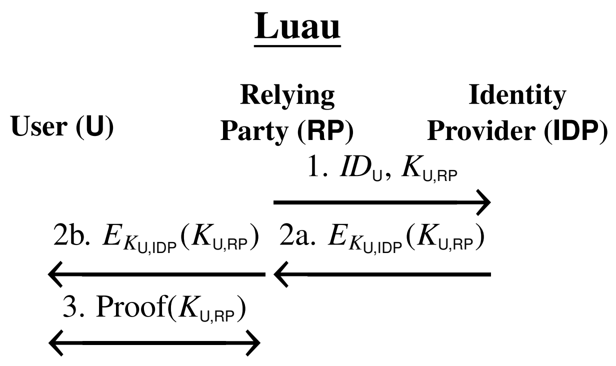
4.2 Description

The *interacting parties* in this protocol are User (U), Relying Party (RP) and Identity Provider (IDP). The flow of messages between these parties at a high level is shown in Fig. 4.1.

> *Goal:* Upon completion of an instance of the protocol, U and RP share an uncompromised session key.

Notation:

$\kappa_{\mathsf{I},\mathsf{J}}$ - A key shared between the principals I and J. It could be any string of finite length.


Luau
User (U)
Relying Party (RP)
Identity Provider (IDP)
1. ID_{U}, $K_{\mathsf{U,RP}}$
2b. $E_{K_{\mathsf{U,IDP}}}(K_{\mathsf{U,RP}})$
2a. $E_{K_{\mathsf{U,IDP}}}(K_{\mathsf{U,RP}})$
3. Proof($K_{\mathsf{U,RP}}$)

Stage 1: After the User initiates the protocol, Relying party and identity provider establish key $K_{\mathsf{U,RP}}$. (The identity provider will only give $K_{\mathsf{U,RP}}$ to the legitimate user identified by ID_{U}.)

Stage 2: The identity provider encrypts $K_{\mathsf{U,RP}}$ such that only the legitimate user can decrypt it. The relying party forwards this package to the user.

Stage 3: The user decrypts $K_{\mathsf{U,RP}}$. Both the user and the relying party then prove knowledge of $K_{\mathsf{U,RP}}$ to each other.

Figure 4.1: High Level Approach of Luau.

$E_{K^{enc}_{\mathsf{I,J}}}(M)$	- Encryption function applied on the message M using the key $K^{enc}_{\mathsf{I,J}}$.
$\text{MAC}_{K^{mac}_{\mathsf{I,J}}}(\text{M})$	- A one-way or irreversible function applied on the message M using the key $K^{mac}_{\mathsf{I,J}}$. This is also called a MAC function or a cryptographic keyed hash function.
$PRF_K(\text{M})$	- A cryptographically secure pseudo-random function with key K and message M.
n_{I}	- A randomly chosen nonce by the principal I. It could be any string of a finite length.

Protocol: The following steps correspond to the messages of the protocol as shown in Fig. 4.2 in detail. An outline of the protocol along with a brief explanation is given in Fig. 4.1.

- Step 1: The user U initiates an instance of the protocol by sending his identity ID_{U} and a nonce n_{U}.
- Step 2: The RP sends its identity ID_{RP} a fresh nonce n_{RP} and it's Diffie-Hellman (DH) public param g^x along with the user's nonce n_{U}.
- Step 3a: This is where a lot of computation is done. The ID_{IDP} generates its own DH param y and uses it to generate $K_{\mathsf{RP,IDP}}$ and in turn uses this key to generate $K^{mac}_{\mathsf{RP,IDP}}$ and $K_{\mathsf{U,RP}}$. It then

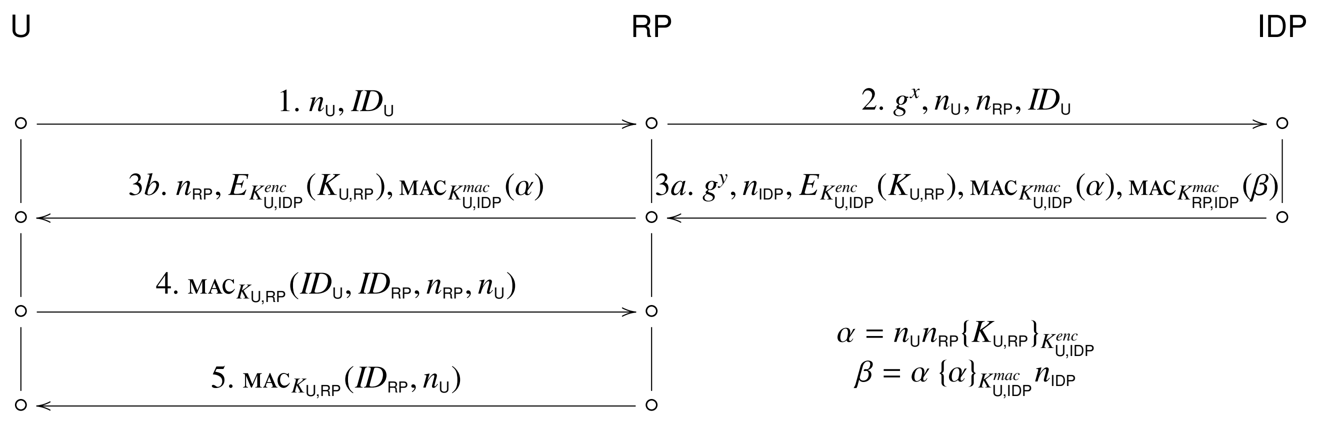

U
RP
IDP
1. $n_{\mathsf{U}}, ID_{\mathsf{U}}$
2. $g^x, n_{\mathsf{U}}, n_{\mathsf{RP}}, ID_{\mathsf{U}}$
3b. $n_{\mathsf{RP}}, E_{K^{enc}_{\mathsf{U,IDP}}}(K_{\mathsf{U,RP}}), \text{MAC}_{K^{mac}_{\mathsf{U,IDP}}}(\alpha)$
3a. $g^y, n_{\mathsf{IDP}}, E_{K^{enc}_{\mathsf{U,IDP}}}(K_{\mathsf{U,RP}}), \text{MAC}_{K^{mac}_{\mathsf{U,IDP}}}(\alpha), \text{MAC}_{K^{mac}_{\mathsf{RP,IDP}}}(\beta)$
4. $\text{MAC}_{K_{\mathsf{U,RP}}}(ID_{\mathsf{U}}, ID_{\mathsf{RP}}, n_{\mathsf{RP}}, n_{\mathsf{U}})$
5. $\text{MAC}_{K_{\mathsf{U,RP}}}(ID_{\mathsf{RP}}, n_{\mathsf{U}})$
$\alpha = n_{\mathsf{U}} n_{\mathsf{RP}} \{K_{\mathsf{U,RP}}\}_{K^{enc}_{\mathsf{U,IDP}}}$
$\beta = \alpha \ \{\alpha\}_{K^{mac}_{\mathsf{U,IDP}}} n_{\mathsf{IDP}}$

Figure 4.2: Luau Protocol

proceeds to send back to RP the public param g^y, a fresh nonce n_{IDP} a *MAC* of β using the key $K^{mac}_{\mathsf{RP,IDP}}$. It also sends the two terms $E_{K^{enc}_{\mathsf{U,IDP}}}(K_{\mathsf{U,RP}})$, $\text{MAC}_{K^{mac}_{\mathsf{U,IDP}}}(\alpha)$ to be forwarded to U without any changes. RP can not tamper with these messages as it does not have access to the keys $K^{enc}_{\mathsf{U,IDP}}$ and $K^{mac}_{\mathsf{U,IDP}}$. These keys are derived from a pre-established long term key $K_{\mathsf{U,IDP}}$ between the user and IDP.

- Step 3b: Upon receiving the previous message, RP generates the key $K_{\mathsf{RP,IDP}}$ by using g^y and its own private exponent x. It then verifies $\text{MAC}_{K^{mac}_{\mathsf{RP,IDP}}}(\beta)$ using the MAC component of $K_{\mathsf{RP,IDP}}$. RP simply forwards the remaining two messages to U and also adds its nonce n_{RP}.

- Step 4: The user U extracts $K_{\mathsf{U,RP}}$ from $E_{K^{enc}_{\mathsf{U,IDP}}}(K_{\mathsf{U,RP}})$ and verifies $\text{MAC}_{K^{mac}_{\mathsf{U,IDP}}}(\alpha)$. This proves that it has the right key that was generated by IDP. It then generates a *MAC* of (ID_{U},ID_{RP},n_{RP},n_{U}) using the key that it extracted and sends the generated $\text{MAC}_{K_{\mathsf{U,RP}}}(ID_{\mathsf{U}},ID_{\mathsf{RP}},n_{\mathsf{RP}},n_{\mathsf{U}})$ to RP. This proves to RP that U possesses the session key $K_{\mathsf{U,RP}}$.

- Step 5: The relying party RP proves its identity by sending a *MAC* of (ID_{RP}, n_{U}) using the key $K_{\mathsf{U,RP}}$ back to U.

$$\begin{array}{rrrr}
 & \mathrm{U}[\mathrm{K}_{(\mathrm{U,IDP})}, \mathrm{K}_{\mathrm{tls(U,RP)}}, & \mathrm{RP}[\mathrm{K}_{\mathrm{tls(U,RP)}}, \mathrm{K}_{\mathrm{tls(RP,IDP)}}, & \mathrm{IDP}[\mathrm{K}_{\mathrm{tls(RP,IDP)}}, \mathrm{K}_{(\mathrm{U,IDP})}, \mathrm{N}_{\mathrm{IDP1}}, \\
 & ID_{\mathrm{RP}}, N_{\mathrm{U}}] & N_{\mathrm{RP1}}, N_{\mathrm{RP2}}] & N_{\mathrm{IDP2}}, K_{(\mathrm{U,RP})}] \\
 & & & \\
1 & & \langle\ -\{\mathrm{K}_{(\mathrm{IDP,RP})}\}_{\mathrm{K}_{\mathrm{tls(RP,IDP)}}} & \langle\ +\{\mathrm{K}_{(\mathrm{IDP,RP})}\}_{\mathrm{K}_{\mathrm{tls(RP,IDP)}}} \\
2 & & -\{\mathrm{N}_{\mathrm{IDP1}} \circ \mathrm{K}_{(\mathrm{U,RP})}\}_{\mathrm{K}_{(\mathrm{IDP,RP})}} & +\{\mathrm{N}_{\mathrm{IDP1}} \circ \mathrm{K}_{(\mathrm{U,RP})}\}_{\mathrm{K}_{(\mathrm{IDP,RP})}} \\
3 & \langle\ -\{\mathrm{N}_{\mathrm{IDP2}} \circ \mathrm{K}_{(\mathrm{U,RP})}\}_{\mathrm{K}_{(\mathrm{U,IDP})}} & & +\{\mathrm{N}_{\mathrm{IDP2}} \circ \mathrm{K}_{(\mathrm{U,RP})}\}_{\mathrm{K}_{(\mathrm{U,IDP})}}\ \rangle \\
4 & +\{\mathrm{ID}_{\mathrm{RP}}\}_{\mathrm{K}_{\mathrm{tls(U,RP)}}} & -\{\mathrm{ID}_{\mathrm{RP}}\}_{\mathrm{K}_{\mathrm{tls(U,RP)}}} & \\
5 & -\{\mathrm{N}_{\mathrm{RP1}},\mathrm{ID}_{\mathrm{RP}}\}_{\mathrm{K}_{\mathrm{tls(U,RP)}}} & +\{\mathrm{N}_{\mathrm{RP1}},\mathrm{ID}_{\mathrm{RP}}\}_{\mathrm{K}_{\mathrm{tls(U,RP)}}} & \\
6 & -\{\mathrm{N}_{\mathrm{RP2}}\}_{\mathrm{K}_{(\mathrm{U,RP})}} & +\{\mathrm{N}_{\mathrm{RP2}}\}_{\mathrm{K}_{(\mathrm{U,RP})}} & \\
7 & +\{\mathrm{N}_{\mathrm{U}} \circ \mathrm{N}_{\mathrm{RP2}}\}_{\mathrm{K}_{(\mathrm{U,RP})}}\ \rangle & -\{\mathrm{N}_{\mathrm{U}} \circ \mathrm{N}_{\mathrm{RP2}}\}_{\mathrm{K}_{(\mathrm{U,RP})}}\ \rangle &
\end{array}$$

Figure 4.3: Luau Strands

4.3 Formal Analysis

Figure 4.3 presents the regular strand traces for Luau strand spaces. The parameters on each trace specify the initial knowledge of the strand, rather than mathematical place holders for all values the trace references. A strand space for Luau is a union of one of each of these strand traces and a set of penetrator strands, such that each identifier's value is distinct, i.e., $\mathrm{K}_{(\mathrm{U,IDP})} \neq \mathrm{ID}_{\mathrm{RP}} \neq \mathrm{N}_{\mathrm{RP1}} \neq \ldots$, and each key is a symmetric key, i.e., $\mathrm{K}_{(\mathrm{U,IDP})}{}^{-1} = \mathrm{K}_{(\mathrm{U,IDP})}$, $\mathrm{K}_{\mathrm{tls(RP,IDP)}}{}^{-1} = \mathrm{K}_{\mathrm{tls(RP,IDP)}}$, etc.

The Strand Space model of Luau is obviously different than the version of Luau presented in Figure 4.2. Many of these differences have to do with the peculiarities of the Strand Space model, while others make explicit the deployment considerations we will argue are important.

First, since the key exchange between U and IDP is not specified by Luau, it is not appropriate to model it in strands. Therefore, we assume it has already occurred. This assumption is reflected in the shared $\mathrm{K}_{(\mathrm{U,IDP})}$ parameter of U and IDP.

Second, since the Strand Space model only handles ideal cryptography, it is not possible to model the key exchange between RP and IDP (messages 2 and 3a of Luau) directly because it relies on Diffie-Hellman, which fabricates a key out of components. Therefore, we model it as IDP unilaterally sending the shared key $\mathrm{K}_{(\mathrm{IDP,RP})}$ to RP.

Third, since the Strand Space model assumes that messages are “abstract” (i.e. they are not bit strings that must be interpreted and may be manipulated in arbitrary ways by the penetrator), we do not include the MACs from the 3-party key exchange (messages 3a and 3b of Luau) because they only serve to verify that the encrypted message component is in pristine condition upon delivery.

Fourth, since the Strand Space model does not consider message transmission to be “directed” (i.e. they are merely sent and received, not sent “to” or received “from”), we model the 3-party key exchange as the IDP unilaterally sending the fresh key to RP then to U, rather than sending both to RP for forwarding to U.

Fifth, we model the mutual authentication of U and RP (messages 4 and 5 of Luau) by simple outgoing authentication tests.

U’s identification of RP (nodes 4 and 5) uses the $K_{tls(U,RP)}$ key and a nonce, ID_{RP}. The nonce is named ID_{RP} to suggest that it describes U’s expectation of the identity of RP, which it must then verify. This test is not part of the high-level account of Luau, but is included to make a practical attack vector explicit.

RP’s identification of U (nodes 6 and 7) uses the $K_{(U,RP)}$ key and a nonce, N_{RP2}. This test corresponds to the mutual authentication step of Luau. The Strand Space model has no separate concept of a “MAC”, so we represent it with encryption.

Finally, we are interested in studying Luau’s security properties in several common deployment scenarios that differ in their use of TLS between various parties. Rather than creating a different version of Luau for each scenario, we use a single Luau model that always uses TLS keys. (See nodes 1, 4 and 5.) We model each different scenario with the availability of these keys to the penetrator. For example, if the penetrator possesses $K_{tls(RP,IDP)}$, then there is no TLS connection between the RP and the IDP.

These TLS keys are not used in every message. Instead, they are only used when a value would otherwise be sent in the clear. For example, the Diffie-Hellman exchange modeled by node 1 normally occurs in the clear, so it is wrapped in the TLS key for RP and IDP. The identification of RP by U (nodes 4 and 5) would also occur in the clear, so it uses the TLS key for U and RP.

4.3.1 Authentication Tests

First, we apply authentication tests to Luau to understand the conditions whereby each party knows that they are communicating with the regular strand we hope they communicate with.

Lemma 1. *If $K_{tls(RP,IDP)} \notin \mathsf{P}$, then node 1 of RP is an unsolicited test for $t = \{K_{(IDP,RP)}\}_{K_{tls(RP,IDP)}}$ where m = node 1 of IDP and $a = K_{(IDP,RP)}$.*

Proof. m is the only positive regular node where t is a component of m. □

Lemma 2. *If $K_{(IDP,RP)} \notin \mathsf{P}$, then node 2 of RP is an unsolicited test for $t = \{N_{IDP1} \circ K_{(U,RP)}\}_{K_{(IDP,RP)}}$ where m = node 2 of IDP and $a = K_{(U,RP)}$.*

Proof. m is the only positive regular node where t is a component of m. □

Lemma 3. *If $K_{(U,IDP)} \notin \mathsf{P}$, then node 3 of U is an unsolicited test for $t = \{N_{IDP2} \circ K_{(U,RP)}\}_{K_{(U,IDP)}}$ where m = node 3 of IDP and $a = K_{(U,RP)}$.*

Proof. m is the only positive regular node where t is a component of m. □

Lemma 4. *If $K_{tls(U,RP)} \notin \mathsf{P}$, then node 4 of RP is an unsolicited test for $t = \{ID_{RP}\}_{K_{tls(U,RP)}}$ where m = node 4 of U and $a = ID_{RP}$.*

Proof. m is the only positive regular node where t is a component of m. □

Lemma 5. *If $K_{tls(U,RP)} \notin \mathsf{P}$, then $n \Rightarrow n'$ is an outgoing authentication test for a in t where n = node 4 of U, n' = node 5 of U, $a = ID_{RP}$, $t = \{ID_{RP}\}_{K_{tls(U,RP)}}$, m = node 4 of RP, m' = node 5 of RP, and m'' = node 5 of U.*

Proof. $\mathrm{ID_{RP}}$ uniquely originates at n. $m \Rightarrow m'$ is the only transforming edge for a. □

Lemma 6. *If $K_{(U,RP)} \notin \mathsf{P}$, then node 6 of U is an unsolicited test for $t = \{N_{RP2}\}_{K_{(U,RP)}}$ where m = node 6 of RP and $a = N_{RP2}$.*

Proof. m is the only positive regular node where t is a component of m. □

Lemma 7. *If $K_{(U,RP)} \notin \mathsf{P}$, then $n \Rightarrow n'$ is an outgoing authentication test for a in t where n = node 6 of RP, n' = node 7 of RP, $a = N_{RP2}$, $t = \{N_{RP2}\}_{K_{(U,RP)}}$, m = node 6 of U, m' = node 7 of U, and m'' = node 7 of RP.*

Proof. N_{RP2} uniquely originates at n. $m \Rightarrow m'$ is the only transforming edge for a. □

4.3.2 Penetrator Analysis

Second, we consider the possible penetrators that can forge messages to the regular nodes and whether this is enough to totally impersonate a particular role.

Definition 27. *The penetrator can impersonate X to Y iff the penetrator can produce every message Y expects from X.*

For example, the penetrator can only impersonate RP to U if it can produce a message that will be accepted by nodes 5 and 6 of the U strand.

Lemma 8. *If $K_{tls(RP,IDP)} \in \mathsf{P}$, then node 1 of RP* **can** *accept a message from a penetrator.*

Proof. The penetrator strands **K**($K_{tls(RP,IDP)}$) and **E**($K_{tls(RP,IDP)}$, K_{P1}) can be used to fabricate a message for node 1 of RP. □

Lemma 9. *If $K_{tls(RP,IDP)} \in \mathsf{P}$, then node 2 of RP* **can** *accept a message from a penetrator.*

Proof. Apply Lemma 8. The penetrator strands **M**(N_P), **K**(K_{P2}), **C**(N_P, K_{P2}), and **E**(K_{P1}, $N_P \circ K_{P2}$) can be used to fabricate a message that node 2 of RP will accept. □

Theorem 3. *If $K_{tls(RP,IDP)} \in \mathsf{P}$, then IDP* **can** *be impersonated to RP by a penetrator.*

Proof. Apply Lemmas 8 and 9. □

Lemma 10. *If $K_{tls(U,RP)} \in \mathsf{P}$, then node 4 of RP* **can** *accept a message from a penetrator.*

Proof. The penetrator strands **M**(ID_P) and **E**($K_{tls(U,RP)}$, ID_P) can fabricate a message that node 4 of RP will accept. □

Lemma 11. *If $K_{tls(U,RP)} \in \mathsf{P}$, then node 5 of U* **can** *accept a message from a penetrator.*

Proof. The penetrator strands $\mathbf{K}(K_{tls(U,RP)})$ and $\mathbf{D}(K_{tls(U,RP)}, ID_{RP})$ can be used to allow the penetrator to learn ID_{RP} from node 4 of U.

Then, the penetrator strands $\mathbf{M}(N_P)$, $\mathbf{C}(N_P, ID_{RP})$, and $\mathbf{E}(K_{tls(U,RP)}, N_P \circ ID_{RP})$ can then fabricate a message that node 5 of U will accept. □

Lemma 12. *If $K_{tls(RP,IDP)} \in \mathsf{P}$, then node 6 of U* **can** *accept a message from a penetrator.*

Proof. The penetrator strands $\mathbf{K}(K_{tls(RP,IDP)})$ and $\mathbf{D}(K_{tls(RP,IDP)}, K_{(IDP,RP)})$ allow the penetrator to learn $K_{(IDP,RP)}$.

The penetrator strands $\mathbf{K}(K_{(IDP,RP)})$, $\mathbf{D}(K_{(IDP,RP)}, m)$, and $\mathbf{S}(m, N_{IDP1}, K_{(U,RP)})$ allow the penetrator to learn $K_{(U,RP)}$.

The penetrator strands $\mathbf{K}(K_{(U,RP)})$, $\mathbf{M}(N_P)$, and $\mathbf{E}(K_{(U,RP)}, N_P)$ produce a message that node 6 of U will accept. □

Theorem 4. *If $\{K_{tls(U,RP)}, K_{tls(RP,IDP)}\} \subseteq \mathsf{P}$, then RP* **can** *be impersonated to U by a penetrator.*

Proof. Apply Lemma 11 and 12. □

Lemma 13. *If $K_{tls(RP,IDP)} \in \mathsf{P}$, then node 7 of RP* **can** *accept a message from a penetrator.*

Proof. By Theorem 3, the penetrator can force RP to use K_P as $K_{(U,RP)}$ for its run.

Therefore, the penetrator strand $\mathbf{D}(K_P, N_{RP2})$ can be used to allow the penetrator to learn N_{RP2} from node 6 of RP.

Then, the penetrator strands $\mathbf{M}(N_P)$, $\mathbf{C}(N_P, N_{RP2})$, and $\mathbf{E}(K_P, N_P \circ N_{RP2})$ can then fabricate a message for node 7 of RP. □

Theorem 5. *If $\{K_{tls(U,RP)}, K_{tls(RP,IDP)}\} \subseteq \mathsf{P}$, then U* **can** *be impersonated to RP by a penetrator.*

Proof. Apply Lemma 10 and 13. □

4.3.3 Scenarios

We now consider four different deployment scenarios and consider the impersonation opportunities of the penetrator. Each scenario will have a table presenting our conclusions. The impersonated party will be written to the left and the gullible party is written above. Combinations that do not interact cannot be in the impersonation relation, by definition, so their cells are blank.

No TLS In this scenario we assume that $\mathsf{P}_0 = \{K_{\mathrm{tls(U,RP)}}, K_{\mathrm{tls(RP,IDP)}}\}$. $\mathsf{P}_1 = \mathsf{P}_0 \cup \{K_{\mathrm{(IDP,RP)}}\}$, because of node 1 of IDP. $\mathsf{P}_2 = \mathsf{P}_1 \cup \{K_{\mathrm{(U,RP)}}\}$, because of node 2 of IDP. $\mathsf{P} = \mathsf{P}_2$, because $\mathrm{K}_{\mathrm{(U,IDP)}}$ is not transmitted in the clear or encrypted by any key in P_2.

	U	RP	IDP
U		Yes, Theorem 5	
RP	Yes, Theorem 4		
IDP	No, Lemma 3	Yes, Theorem 3	

TLS from (RP → IDP) only In this scenario we assume that $\mathsf{P}_0 = \{K_{\mathrm{tls(U,RP)}}\}$. $\mathsf{P} = \mathsf{P}_0$, because no keys are encrypted with $\mathrm{K}_{\mathrm{tls(U,RP)}}$.

	U	RP	IDP
U		No, Lemma 7	
RP	No[1], Lemma 6		
IDP	No, Lemma 3	No, Lemma 1	

TLS from (U → RP) only In this scenario we assume that $\mathsf{P}_0 = \{K_{\mathrm{tls(RP,IDP)}}\}$. $\mathsf{P}_1 = \mathsf{P}_0 \cup \{K_{\mathrm{(IDP,RP)}}\}$, because of node 1 of IDP. $\mathsf{P}_2 = \mathsf{P}_1 \cup \{K_{\mathrm{(U,RP)}}\}$, because of node 2 of IDP. $\mathsf{P} = \mathsf{P}_2$, because $\mathrm{K}_{\mathrm{tls(U,RP)}}$ and $\mathrm{K}_{\mathrm{(U,IDP)}}$ are not transmitted in the clear or encrypted by any key in P_2.

[1]However, Lemma 11 applies.

	U	RP	IDP
U		No[2], Lemma 4	
RP	No, Lemma 5		
IDP	No, Lemma 3	Yes, Theorem 3	

TLS from (RP → IDP) and (U → RP) In this scenario we assume that $\mathsf{P}_0 = \emptyset$. $\mathsf{P} = \mathsf{P}_0$, because no values are sent in the clear.

	U	RP	IDP
U		No, Lemma 4	
RP	No, Lemma 5		
IDP	No, Lemma 3	No, Lemma 1	

Conclusion This analysis suggests that Luau is most secure when TLS is used between each party, but is quite secure when TLS is only used between the RP and the IDP. The latter is only insecure in a single message to the user verifying that the RP is the expected RP, but since the other message from the RP to the user (node 6) cannot be forged, this implies that the shared key between the user and RP is still safe, so the protocol goal of establishing a secure shared key is fulfilled. In contrast, the other scenarios are quite insecure because they allow the penetrator full control over the shared key.

[2]However, Lemma 13 applies and it is arguably more important from RP's perspective.

Chapter 5

pwdArmor

5.1 Introduction

Traditional authentication systems send passwords as plain text or within an encrypted tunnel. pw-dArmor [11] is an approach to securing passwords from both passive and active attackers without having to change the underlying password verifier database.

Password-based authentication mechanisms over the Internet fall into two categories in the context of password transmission. The password is either transmitted in the clear or it is transmitted via a secure tunnel between the user and the host. The first type offers very minimal security as an eavesdropper can obtain the password while it is in transit. The second type is widely used and offers an acceptable level of protection, but phishing sites can circumvent that protection in a number of ways. One common way is to simply not create the secure tunnel and hope that the users do not notice it. Another way is to trick the user into accepting the wrong certificate.

5.2 Description

pwdArmor prevents passive eavesdropping attacks and also improves the detection of active man-in-the-middle attacks. The protocol is shown in Figure 5.1.

- Step 1: The user U initiates the protocol by sending her identity ID_U and nonce N_U.
- Step 2: The host H responds with its nonce N_H, Diffie-Hellman public key parameter g^x, and the challenge C that is specific to the underlying password protocol.

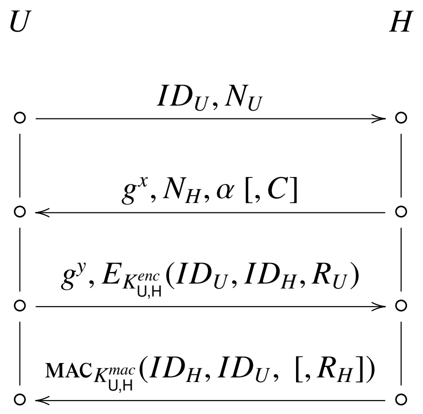

U
H
ID_U, N_U
$g^x, N_H, \alpha\ [, C]$
$g^y, E_{K^{enc}_{\mathsf{U,H}}}(ID_U, ID_H, R_U)$
$\text{MAC}_{K^{mac}_{\mathsf{U,H}}}(ID_H, ID_U,\ [, R_H])$

Figure 5.1: pwdArmor Protocol

- Step 3: The user then responds with her DH public key parameter and an encrypted response which contains the terms ID_U, ID_H and a response R_U. The key used for encryption is derived as follows: $K^{enc}_{\mathsf{U,H}} = PRF_{K_{\mathsf{U,H}}}(pwd^{ver}_U, \text{''}enc\text{''})$ where pwd^{ver}_U is the password in a format verifiable by the host, $K_{\mathsf{U,H}} = PRF_{n_U \| n_H}(g^{xy})$ and PRF is a family of pseudo random functions. This step authenticates the user to the host.
- Step 4: If the user submits the correct password verifier pwd^{ver}_U, the host responds with message 4 demonstrating its knowledge of pwd^{ver}_U where $K^{mac}_{\mathsf{U,H}} = PRF_{K_{\mathsf{U,H}}}(pwd^{ver}_U, \text{''}mac\text{''})$.

5.3 Formal Analysis

Figure 5.2 is the Strand Spaces representation of pwdArmor. This representation makes similar assumptions for the use of TLS keys and Diffie-Hellman keys that were made in the formal analysis of Luau.

The Strand Spaces model of pwdArmor does not directly represent the real world objectives of pwdArmor, which is to strengthen existing password-based protocols. The formal analysis illustrates that pwdArmor does not introduce any additional vulnerabilities into existing password-based protocols.

$$\begin{array}{lrr}
 & U[K_{pwd(U,H)}, K_{tls(U,H)}, ID_H, ID_H, N_U] & H[K_{tls(U,H)}, K_{pwd(U,H)}, N_H\ ] \\
1 & \langle\ +\{K_{(U,H)}\}_{K_{tls(U,H)}} & \langle\ -\{K_{(U,H)}\}_{K_{tls(U,H)}} \\
2 & +\{K_{enc(U,H)}\}_{K_{pwd(U,H)}} & -\{K_{enc(U,H)}\}_{K_{pwd(U,H)}} \\
3 & +\{ID_U \circ ID_H \circ N_U\}_{K_{enc(U,H)}} & -\{ID_U \circ ID_H \circ N_U\}_{K_{enc(U,H)}} \\
4 & -\{K_{mac(U,H)}\}_{K_{pwd(U,H)}} & +\{K_{mac(U,H)}\}_{K_{pwd(U,H)}} \\
5 & -\{ID_U \circ ID_H \circ N_H\}_{K_{mac(U,H)}}\ \rangle & +\{ID_U \circ ID_H \circ N_H\}_{K_{mac(U,H)}}\ \rangle
\end{array}$$

Figure 5.2: pwdArmor Strands

In Figure 5.2, the key $K_{(U,H)}$ represents the key generated using the Diffie-Hellman exchange in the original protocol. In the Strand Spaces model of pwdArmor, this key is is not used in any subsequent steps or to establish any security properties, but is shown just to represent the Diffie-Hellman exchange in the original protocol.

In the original protocol, pwd_U^{ver} is a pre-shared secret between U and H. This is modeled as the pre-shared key $K_{pwd(U,H)}$.

In the original protocol, the keys $K_{U,H}^{enc}$ and $K_{U,H}^{mac}$ are derived using a PRF and other components such as nonces and pwd_U^{ver}. Since we cannot represent key derivation in Strand Spaces, we model this as the keys being securely transmitted between participants encrypted using the pre-shared secret $K_{pwd(U,H)}$.

The nonces N_U and N_H are used for adding freshness to the messages 3 and 5 respectively.

5.3.1 Authentication Tests

Let us now apply authentication tests to this protocol to see under what conditions a principal believes that it is talking to another honest principal and not the penetrator.

Lemma 14. *If $K_{tls(U,H)} \notin \mathcal{P}$, then node 1 of H is an unsolicited test for $t = \{K_{(U,H)}\}_{K_{tls(U,H)}}$ where $m =$ node 1 of U and $a = K_{(U,H)}$.*

Proof. m is the only positive regular node where t is a component of m. □

Lemma 15. *If $K_{pwd(U,H)} \notin \mathsf{P}$, then node 2 of H is an unsolicited test for $t = \{K_{enc(U,H)}\}_{K_{pwd(U,H)}}$ where m = node 2 of U and $a = K_{enc(U,H)}$.*

Proof. m is the only positive regular node where t is a component of m. □

Lemma 16. *If $K_{enc(U,H)} \notin \mathsf{P}$, then node 3 of H is an unsolicited test for $t = \{ID_U \circ ID_H \circ N_U\}_{K_{enc(U,H)}}$ where m = node 3 of U and $a = N_U$.*

Proof. m is the only positive regular node where t is a component of m. □

Lemma 17. *If $K_{pwd(U,H)} \notin \mathsf{P}$, then node 4 of U is an unsolicited test for $t = \{K_{mac(U,H)}\}_{K_{pwd(U,H)}}$ where m = node 4 of H and $a = K_{mac(U,H)}$.*

Proof. m is the only positive regular node where t is a component of m. □

Lemma 18. *If $K_{enc(U,H)} \notin \mathsf{P}$, then $n \Rightarrow n'$ is an outgoing authentication test for a in t where n = node 3 of U, n' = node 5 of U, $a = ID_H$, $t = \{ID_U \circ ID_H \circ N_U\}_{K_{enc(U,H)}}$, m = node 3 of H, m' = node 5 of H, and m'' = node 5 of U.*

Proof. $\mathrm{ID_H}$ uniquely originates at n. $m \Rightarrow m'$ is the only transforming edge for a. □

5.3.2 Penetrator

The goal of the penetrator is to impersonate U to H. This implies, the penetrator should produce every message that H expects from U.

Lemma 19. *If $K_{tls(U,H)} \in \mathsf{P}$, then node 1 of H* **can** *accept a message from a penetrator.*

Proof. The penetrator strands $\mathbf{K}(\mathrm{K_{tls(U,H)}})$ and $\mathbf{E}(\mathrm{K_{tls(U,H)}}, \mathrm{K_{(U,H)}})$ can be used to fabricate a message for node 1 of H. □

This is the only message that the penetrator can fabricate for H since the key $\mathrm{K_{pwd(U,H)}}$ is required to fabricate any subsequent messages, but $K_{\mathrm{pwd(U,H)}} \in S_0$.

For the same reason, the penetrator cannot impersonate H to U.

5.3.3 Scenarios

There are two scenarios mentioned in the threat analysis in [11].

S_{clear}**:** An unsecured channel is used for all communications. This scenario is equivalent to $K_{\mathrm{tls(U,H)}} \in P_0$. This implies $K_{\mathrm{(U,H)}} \in P_1$. But the penetrator cannot impersonate U since Lemma 16 holds true and U is authenticated to H. Similarly, the penetrator cannot impersonate H since Lemma 18 holds true and H is authenticated to U.

S_{tunnel}**:** A server authenticated encrypted tunnel is used for all communications. In this scenario no values are sent in clear. Hence, the same Lemmas apply as above and U and H authenticate to each other securely.

Conclusion: Authentication is successful in both scenarios since the penetrator never knows $\mathrm{K}_{\mathrm{pwd(U,H)}}$ even though he may have access to $\mathrm{K}_{\mathrm{(U,H)}}$ in the first scenario.

Chapter 6

Kiwi

6.1 Introduction

Kiwi is a protocol for distributing symmetric keys between users in the context of end-to-end secure email. The three primary participants involved in this protocol are the Authentication Server, Key Server and the Client. There are two roles for clients: Creator and Viewer. This protocol can be used when there is no prior shared secret between the clients and the keys are distributed by a third party Key Server. The Key Server is designed in such a way that it does not store any keys except its own master key K_{KS}, but instead it generates creator and viewer keys by using the master key. While this protocol had been designed to provide end-to-end secure email without the usability pitfalls of PKI, it has also been implemented for providing a secure storage solution [4].

Simple Authentication for the Web (SAW) [9] is an authentication protocol that provides a single sign-on solution for users and service providers. In this protocol, the user's email provider acts as the identity provider. The service provider splits the authentication token into two halves and sends one half directly to the user and the second half to the email address provided by the user. The user is authenticated if he/she replies with the complete token, thus proving ownership of the email address. SAW can be used as the authentication mechanism in Kiwi.

6.2 Description

Here are the steps involved in the protocol:

- The Client (either creator or viewer) authenticates to the Authentication Server using SAW.

- The Authentication Server generates an authorization token (AT) using the key $K_{(AS,KS)}$ (the shared secret key between Key Server and the Authentication Server), the identity of the client and the time period this token is valid for. $AT \leftarrow KDF_{K_{AS}}(id_U \| \tau)$. This token is sent back to the client.
- The client then sends this token and the time period of its validity to the Key Server and depending on the role, it requests either a creator key or a viewer key. The key server verifies the token by re-creating the token with $K_{(AS,KS)}$ and matching it with what the client had submitted, thereby authenticating the client.
- The creator key is generated as $k_C \leftarrow KDF_{K_{KS}}(id_C \| \tau)$. A viewer key is specific to a creator-viewer pair, so the creator key k_C is generated first and the the viewer key is created as $k_{c,v} \leftarrow KDF_{K_C}(id_v)$.

Figure 6.1 shows the interactions of the creator with the authentication server, key server and the email provider at a high level. At the end of step 3, the creator can generate the authorization token as $AT_C = AT_{C1} + AT_{C2}$. This token is then sent to the key server in step 4 to retrieve the creator key K_C.

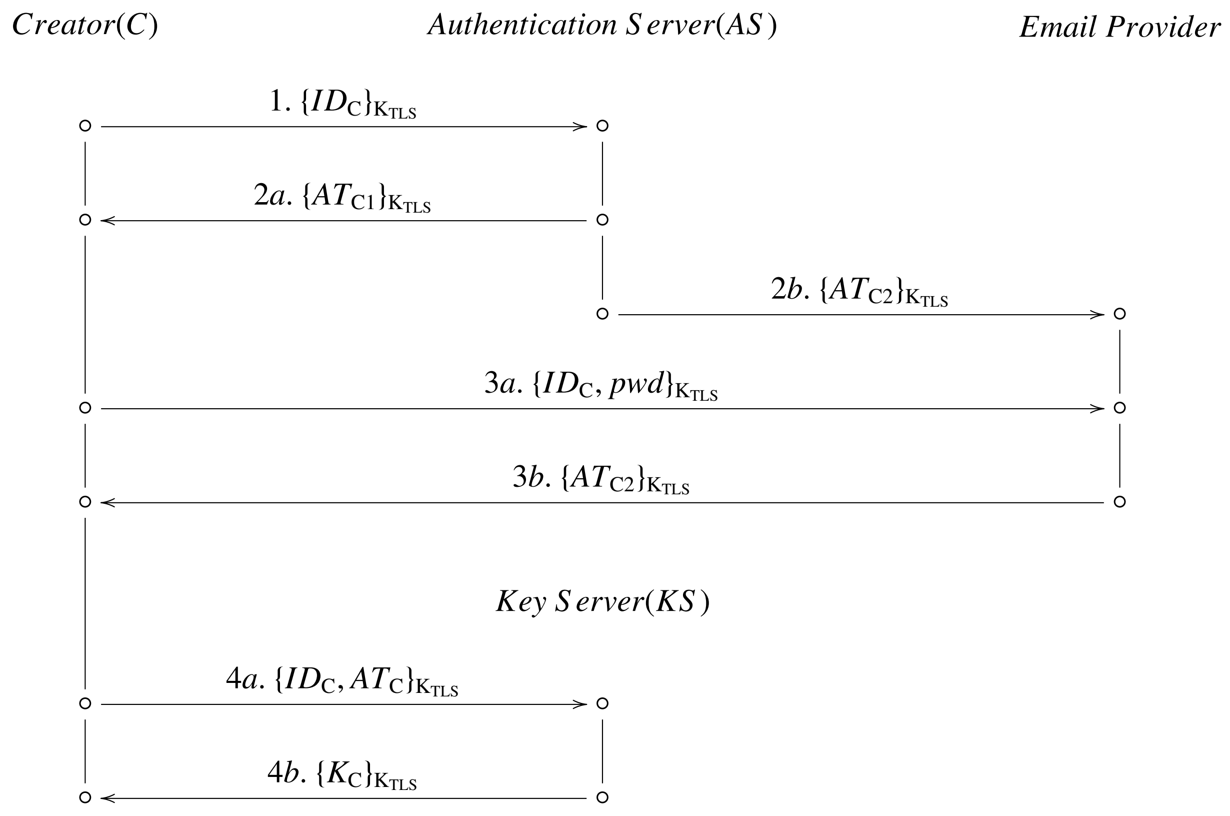
6.3 Formal Analysis

Figure 6.2 shows the Strand Spaces representation of the Kiwi protocol. There are some important differences between the model and the original protocol in Figure 6.1, which are as follows.

In the original protocol, C authenticates to EP by providing a password. We model this as a pre-shared secret key $K_{pwd(C,EP)}$ between C and EP.

We employ the TLS trick to model the TLS connection between entities. The same strategy was applied in the earlier protocols Luau and pwdArmor.

To model token splitting of SAW, we assume that C submits the two halves of the authentication token AT_{C1} and AT_{C2} separately to KS. We then stipulate that C is authenticated by the KS only if both halves are submitted to KS and validated.



$Creator(C)$
$Authentication\ Server(AS)$
$Email\ Provider$
$1.\ \{ID_C\}_{K_{TLS}}$
$2a.\ \{AT_{C1}\}_{K_{TLS}}$
$2b.\ \{AT_{C2}\}_{K_{TLS}}$
$3a.\ \{ID_C, pwd\}_{K_{TLS}}$
$3b.\ \{AT_{C2}\}_{K_{TLS}}$
$Key\ Server(KS)$
$4a.\ \{ID_C, AT_C\}_{K_{TLS}}$
$4b.\ \{K_C\}_{K_{TLS}}$

Figure 6.1: Kiwi Protocol (Creator)

In the original protocol, Authentication Server(AS) and Key Server(KS) are two different entities with different functionalities. KS has access to K_{AS} which is used by AS to generate the Authentication token and KS uses the key to verify it. In Figure 6.2, we model this as a single entity since KS has all the initial knowledge of AS and can essentially impersonate it.

All subsequent keys are derived from the key K_C. Hence, it is sufficient to show that this key is secure in order to prove the secrecy of the remaining keys, assuming ideal cryptography.

6.3.1 Authentication Tests

Let us now identify the various authentication tests present in this protocol.

Lemma 20. *If $K_{tls(C,KS)} \notin P$, then node 1 of C is an unsolicited test for $t = \{AT_{C1}\}_{K_{tls(C,KS)}}$ where $m =$ node 1 of KS and $a = AT_{C1}$.*

	$C[K_{tls(C,KS)}, K_{pwd(C,EP)}, K_{tls(C,EP)}, ID_C]$	$KS[K_{tls(C,KS)}, K_{tls(KS,EP)}, N_1, AT_{C1}, AT_{C2}]$	$EP[K_{tls(KS,EP)}, K_{tls(C,EP)}]$
1	$\langle -\{AT_{C1}\}_{K_{tls(C,KS)}}$	$\langle +\{AT_{C1}\}_{K_{tls(C,KS)}}$	
2		$+\{AT_{C2}\}_{K_{tls(KS,EP)}}$	$\langle -\{AT_{C2}\}_{K_{tls(KS,EP)}}$
3	$+\{ID_C\}_{K_{pwd(C,EP)}}$		$-\{ID_C\}_{K_{pwd(C,EP)}}$
4	$-\{AT_{C2}\}_{K_{tls(C,EP)}}$		$+\{AT_{C2}\}_{K_{tls(C,EP)}} \rangle$
5	$+\{ID_C \circ AT_{C1}\}_{K_{tls(C,KS)}}$	$-\{ID_C \circ AT_{C1}\}_{K_{tls(C,KS)}}$	
6	$-\{N_1\}_{K_{tls(C,KS)}}$	$+\{N_1\}_{K_{tls(C,KS)}}$	
7	$+\{N_1 \circ AT_{C2}\}_{K_{tls(C,KS)}}$	$-\{N_1 \circ AT_{C2}\}_{K_{tls(C,KS)}}$	
8	$-\{K_C\}_{K_{tls(C,KS)}} \rangle$	$+\{K_C\}_{K_{tls(C,KS)}} \rangle$	

Figure 6.2: Kiwi Strands

Proof. m is the only positive regular node where t is a component of m. □

Lemma 21. *If $K_{tls(KS,EP)} \notin P$, then node 2 of EP is an unsolicited test for $t = \{AT_{C2}\}_{K_{tls(KS,EP)}}$ where m = node 2 of KS and $a = AT_{C2}$.*

Proof. m is the only positive regular node where t is a component of m. □

Lemma 22. *If $K_{pwd(C,EP)} \notin P$, then node 3 of EP is an unsolicited test for $t = \{ID_C\}_{K_{pwd(C,EP)}}$ where m = node 3 of C and $a = ID_C$.*

Proof. m is the only positive regular node where t is a component of m. □

Lemma 23. *If $\{K_{tls(C,EP)}, K_{tls(KS,EP)}\} \notin P$, then node 4 of C is an unsolicited test for $t = \{AT_{C2}\}_{K_{tls(C,EP)}}$ where m = node 2 of KS and $a = AT_{C2}$.*

Proof. m is the only positive regular node where t is a component of m. □

Lemma 24. *If $K_{tls(C,KS)} \notin P$, then $n \Rightarrow n'$ is an outgoing authentication test for a in t where n = node 1 of KS, n' = node 5 of KS, $a = AT_{C1}$, $t = \{AT_{C1}\}_{K_{tls(C,KS)}}$, m = node 1 of C, m' = node 5 of C, and m'' = node 5 of KS.*

Proof. AT_{C1} uniquely originates at n. $m \Rightarrow m'$ is the only transforming edge for a. □

Lemma 25. *If $K_{tls(KS,EP)} \notin \mathsf{P}$, then $n \Rightarrow n'$ is an outgoing authentication test for a in t where n = node 2 of KS, n' = node 7 of KS, $a = AT_{C2}$, $t = \{AT_{C2}\}_{K_{tls(KS,EP)}}$, m = node 2 of EP, m' = node 4 of EP, and m'' = node 4 of C .*

Proof. AT_{C2} uniquely originates at n. $m \Rightarrow m'$ is the transforming edge for a. □

6.3.2 Penetrator

The goal of the penetrator is to get access to K_C either by passive or active means. This could mean that the penetrator could impersonate the Creator and fool the Key Server into giving it the key, or impersonate the Key Server and submit its own key to the Creator. Let's see under what conditions the penetrator can impersonate each of the entities.

Lemma 26. *If $K_{tls(C,KS)} \in \mathsf{P}$, then node 1 of C* **can** *accept a message from a penetrator.*

Proof. The penetrator strands $\mathbf{K}(K_{tls(C,KS)})$, $\mathbf{M}(AT_{P1})$ and $\mathbf{E}(K_{tls(C,KS)}, AT_{P1})$ can be used to fabricate a message for node 1 of C. □

Lemma 27. *If $K_{tls(KS,EP)} \in \mathsf{P}$, then node 2 of EP* **can** *accept a message from a penetrator.*

Proof. The penetrator strands $\mathbf{K}(K_{tls(KS,EP)})$, $\mathbf{M}(AT_{P2})$ and $\mathbf{E}(K_{tls(KS,EP)}, AT_{P1})$ can be used to fabricate a message for node 2 of EP. □

Lemma 28. *If $K_{tls(C,EP)} \in \mathsf{P}$, then node 4 of C* **can** *accept a message from a penetrator.*

Proof. The penetrator strands $\mathbf{K}(K_{tls(C,EP)})$, $\mathbf{M}(AT_{P2})$ and $\mathbf{E}(K_{tls(C,EP)}, AT_{P2})$ can be used to fabricate a message for node 4 of C. □

Lemma 29. *If $K_{tls(C,KS)} \in \mathsf{P}$, then node 6 of C* **can** *accept a message from a penetrator.*

Proof. Apply Lemma 26. The penetrator strands $\mathbf{M}(N_P)$, $\mathbf{E}(K_{tls(C,KS)},N_P)$ can be used to fabricate a message that node 6 of C will accept. □

Lemma 30. *If $K_{tls(C,KS)} \in \mathsf{P}$, then node 8 of C* **can** *accept a message from a penetrator.*

Proof. Apply Lemma 26 and Lemma 29. The penetrator strands **K**(K_{P1}), **E**($K_{tls(C,KS)}$,K_{P1}) can be used to fabricate a message that node 8 of C will accept. □

Theorem 6. *If* $\{K_{tls(C,KS)}, K_{tls(KS,EP)}\} \subseteq P$*, then KS* **can** *be impersonated to C by a penetrator.*

Proof. Apply Lemmas 26, 27, 29 and 30. □

Theorem 7. *If* $\{K_{tls(C,KS)}, K_{tls(C,EP)}\} \subseteq P$*, then KS* **can** *be impersonated to C by a penetrator.*

Proof. Apply Lemmas 26, 28, 29 and 30. □

Lemma 31. *If* $K_{tls(C,KS)} \in P$*, then node 5 of KS* **can** *accept a message from a penetrator.*

Proof. The penetrator strands **K**($K_{tls(C,KS)}$) and **D**($K_{tls(C,KS)}$, AT_{C1}) can be used to allow the penetrator to learn AT_{C1} from node 1 of KS.

Then, penetrator strands **M**(ID_C), **C**(ID_C, AT_{C1}) and **E**($K_{tls(C,KS)}$, $ID_C \circ AT_{C1}$) can be used to fabricate a message for node 5 of KS. □

Lemma 32. *If* $\{K_{tls(C,KS)},K_{tls(KS,EP)}\} \in P$*, then node 7 of KS* **can** *accept a message from a penetrator.*

Proof. The penetrator strands **K**($K_{tls(KS,EP)}$) and **D**($K_{tls(KS,EP)}$, AT_{C2}) can be used to allow the penetrator to learn AT_{C2} from node 2 of KS.

The penetrator strands **K**($K_{tls(C,KS)}$) and **D**($K_{tls(C,KS)}$, N_1) can be used to allow the penetrator to learn N_1 from node 6 of KS.

Then, penetrator strands **C**(N_1, AT_{C2}) and **E**($K_{tls(C,KS)}$, $N_1 \circ AT_{C2}$) can be used to fabricate a message for node 7 of KS. □

Theorem 8. *If* $\{K_{tls(C,KS)}, K_{tls(KS,EP)}\} \subseteq P$*, then C* **can** *be impersonated to KS by a penetrator.*

Proof. Apply Lemmas 31 and 32. □

Lemma 33. *If* $\{K_{tls(C,KS)},K_{tls(C,EP)}\} \in P$*, then node 7 of KS* **can** *accept a message from a penetrator.*

Proof. The penetrator strands **K**($K_{tls(C,EP)}$) and **D**($K_{tls(C,EP)}$, AT_{C2}) can be used to allow the penetrator to learn AT_{C2} from node 4 of EP.

The penetrator strands **K**($K_{tls(C,KS)}$) and **D**($K_{tls(C,KS)}$, N_1) can be used to allow the penetrator to learn N_1 from node 6 of KS.

Then, penetrator strands **C**(N_1, AT_{C2}) and **E**($K_{tls(C,KS)}$, $N_1 \circ AT_{C2}$) can be used to fabricate a message for node 7 of KS. □

Theorem 9. *If* $\{K_{tls(C,KS)}, K_{tls(C,EP)}\} \subseteq P$*, then C* **can** *be impersonated to KS by a penetrator.*

Proof. Apply Lemmas 31 and 33. □

Lemma 34. *If* $K_{tls(KS,EP)} \in P$*, then node 2 of EP* **can** *accept a message from a penetrator.*

Proof. The penetrator strands **K**($K_{tls(KS,EP)}$), **M**(AT_{P2}) and **E**($K_{tls(C,KS)}$, AT_{P2}) can be used to fabricate a message for node 2 of EP. □

Theorem 10. *If* $K_{tls(KS,EP)} \in P$*, then KS* **can** *be impersonated to EP by a penetrator.*

Proof. Apply Lemma 34. □

Theorem 11. *If* $K_{tls(C,EP)} \in P$*, then EP* **can** *be impersonated to C by a penetrator.*

Proof. Apply Lemma 28. □

Lemma 35. *If* $K_{pwd(C,EP)} \in P$*, then node 3 of EP* **can** *accept a message from a penetrator.*

Proof. The penetrator strands **K**($K_{pwd(C,EP)}$), **M**(ID_C) and **E**($K_{pwd(C,EP)}$, ID_C) can be used to fabricate a message for node 3 of EP. □

Theorem 12. *If* $K_{pwd(C,EP)} \in P$*, then C* **can** *be impersonated to EP by a penetrator.*

Proof. Apply Lemma 35. □

6.3.3 Scenarios

Let us now look at different scenarios and see whether the protocol is secure in each one of them. We present these in a tabular form as presented previously in 4.3.3.

When we say that a particular TLS key is known to the penetrator, it implies that the key is compromised. This could happen when the penetrator initiates a TLS session with the other principal without being authenticated, or it could also denote the absence of a TLS connection between those principals.

We always assume that there is a TLS connection C and EP as it is one of the requirements of the original protocol. The scenarios show why it is always a good idea to have this connection.

In the original protocol, when a TLS connection is established between C and AS, only the AS is authenticated but not C. The scenarios which show this connection being compromised are important since they reflect the conditions in the original protocol.

We do not show the scenarios where there is no TLS connection between any of the entities since that is inherently insecure and also the one where there is a TLS connection between all three entities since that is most secure.

Note: In the original protocol, KS does not authenticate EP. Hence, we do not need to specifically show that EP can be impersonated to KS.

TLS from (C $\rightarrow$ EP) only In this scenario we assume that $\mathsf{P}_0 = \{K_{\text{tls(C,KS)}}, K_{\text{tls(KS,EP)}}\}$. $\mathsf{P}_1 = \mathsf{P}_0 \cup \{K_{\mathrm{C}}\}$, because of node 8 of KS. All subsequent keys are derived from the key K_{C}. Hence, they also belong to the set of penetrator keys.

The assumption $K_{\text{tls(C,KS)}} \in \mathsf{P}$ implies that KS established the TLS connection with the penetrator and not with a regular principal.

The assumption $K_{\text{tls(KS,EP)}} \in \mathsf{P}$ implies that there is no TLS connection between KS and EP. This is the default scenario as outlined in the threat analysis section of SAW [9]. In the paper, this is considered to be an acceptable risk.

	C	KS	EP
C		Yes, Theorem 8	No, Lemma 22
KS	Yes, Theorem 6		Yes, Theorem 10
EP	No, Lemma 23		

TLS from (C → KS) only In this scenario, we assume that C and KS are both honest participants and there is a secure TLS connection between them. Hence, $\mathsf{P}_0 = \{K_{\text{tls(KS,EP)}}, K_{\text{tls(C,EP)}}\}$.

	C	KS	EP
C		No, Lemma 24	Yes, Theorem 12
KS	No, Lemma 20		Yes, Theorem 10
EP	Yes, Lemma 28		

TLS from (KS → EP) only In this scenario, we assume that only KS and EP share a secure TLS connection. Hence, $\mathsf{P}_0 = \{K_{\text{tls(C,KS)}}, K_{\text{tls(C,EP)}}\}$.

	C	KS	EP
C		Yes, Theorem 9	No, Lemma 22
KS	Yes, Theorem 7		No, Lemma 21
EP	Yes, Theorem 12		

TLS from (KS → EP) and (C → EP) only In this scenario we assume that there is no TLS connection between C and KS. Hence, $\mathsf{P}_0 = \{K_{\text{tls(C,KS)}}\}$ only. It does not imply that there is no TLS connection between C and KS, but instead only that the penetrator is trying to impersonate either of the two principals. So there is a TLS connection but it is compromised.

	C	KS	EP
C		No, Lemma 24	No, Lemma 22
KS	No, Lemma 23		No, Lemma 21
EP	No, Lemma 23		

TLS from (KS → EP) and (C → KS) only In this scenario, there is no TLS connection between C and EP. Hence, $P_0 = \{K_{tls(C,KS)}\}$ only.

	C	KS	EP
C		No, Lemma 24	Yes, Theorem 12
KS	No, Lemma 23		No, Lemma 21
EP	Yes, Theorem 12		

TLS from (C → EP) and (C → KS) only In this scenario, there is no TLS connection between KS and EP. Hence, $P_0 = \{K_{tls(KS,EP)}\}$ only. Although the analysis shows that KS can be impersonated to EP, this is not of much use since the penetrator does not gain any valuable information as the protocol progresses.

	C	KS	EP
C		No, Lemma 24	No, Lemma 22
KS	No, Lemma 23		Yes, Theorem 10
EP	No, Lemma 23		

Conclusion The most secure of the scenarios are those where $K_{tls(C,EP)}$ is not compromised. Hence it validates the requirement in the original protocol that there should always be an uncompromised TLS connection between C and EP.

The analysis shows that the protocol is most secure when there is an uncompromised TLS connection between KS and EP and between C and EP. In the original protocol, this translates to a tunnel between the Authentication Server and the Email Provider and between the Client and the Email Provider.

Essentially, the protocol is secure if either of the two Authentication tokens are secure. In the original protocol, we cannot ensure the secrecy of AT_1, but scenario "TLS from (KS → EP) and (C → EP) only" ensures that AT_2 is secure.

Chapter 7

Conclusions

This thesis provides a formal analysis of the correctness of the protocols Luau, pwdArmor and Kiwi. This analysis provides increased assurance of the security properties that the protocols claim to provide. We evaluate each protocol by providing scenarios where we analyze the security of the protocol in the presence or absence of a TLS connection between interacting principals.

Luau is a 3-party secure key distribution protocol that leverages the implicit trust that service providers place in identity providers. We show that the Luau protocol is still highly secure when there is a TLS connection only between RP and IDP. This is in line with the informal analysis presented in [10].

pwdArmor is used to strengthen existing password based authentication protocols without changing the password verifier database. We show that the pwdArmor protocol does not introduce any additional vulnerabilities and the password verifier is secure both in the presence and absence of a TLS connection.

Kiwi is a symmetric key distribution protocol, where a Key Server distributes encryption keys to participants who want to securely communicate with each other. An Authentication Server is responsible for authenticating the participants before any key distribution takes place. SAW is used as the authentication protocol here. We conclude that the Kiwi protocol is secure even if the Penetrator has access to either one of the two Authentication Tokens but not both.

References

[1] Michael Burrows, Martn Abadi, and Roger Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8:18–36, 1990.

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

[3] F. Javier Thayer Fabrega, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Why is a security protocol correct? In *Proceedings IEEE Symposium on Security and Privacy*, May 1998.

[4] Trevor Florence. Kiwidrive: A portable encryption solution for usb flash drives. Master's thesis, Brigham Young University, 2009.

[5] Gavin Lowe. An attack on the needham-schroeder public key authentication protocol. *Information Processing Letters*, 56(3):133–136, 1995.

[6] Catherine Meadows. The nrl protocol analyzer: An overview. *The Journal of Logic Programming*, 26(2):113 – 131, 1996.

[7] Roger Needham and Michael Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12), 1978.

[8] Peter Ryan and Steve Schneider. *The Modelling and Analysis of Security Protocols: the CSP Approach*. Addison-Wesley, 2001.

[9] Timothy W. van der Horst and Kent E. Seamons. Simple authentication for the web. In *Third International Conference on Security and Privacy in Communications Networks and the Workshops, SecureComm*, pages 473–482, Nice, France, 2007.

[10] Timothy W. van der Horst, Pavan Vankamamidi, Jay McCarthy, and Kent E. Seamons. Luau user authentication. Under submission, 2011.

[11] T.W. van der Horst and K.E. Seamons. pwdarmor: Protecting conventional password-based authentications. In *ACSAC 2008: Annual Computer Security Applications Conference*, pages 443 –452, December 2008.