Specifying and Modelling Secure Channels in Strand Spaces

Allaa Kamil and Gavin Lowe

Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford, OX1 3QD, UK
{allaa.kamil,gavin.lowe}@comlab.ox.ac.uk

Abstract. We adapt the Strand Spaces model to reason abstractly about layered security protocols, where an Application Layer protocol is layered on top of a secure transport protocol. The model abstracts away from the implementation of the secure transport protocol and just captures the properties that it provides to the Application Layer. We illustrate the usefulness of the model by using it to verify a small single sign-on protocol.

1 Introduction

Many security architectures make use of layering of protocols: a special-purpose *Application Layer* protocol is layered on top of a general-purpose *Secure Transport Layer* protocol, such as SSL/TLS [Tho00]. The secure transport protocol provides a *secure channel* to the Application Layer, i.e., it provides a communication channel with some extra security services such as authentication and confidentiality. The Application Layer protocol builds on this to provide extra functionality and security guarantees.

As an example, one common use of such layered architectures is in Single-Sign-On (SSO) protocols. In such protocols, a *User* seeks to access services provided by a *Service Provider*; the User is authenticated by a trusted *Identity Provider*. Typically, the User can open a unilateral TLS connection to the Service Provider, which authenticates the Service Provider but not the User. Further, the User can open a unilateral TLS connection to the Identity Provider, which authenticates the Identity Provider; the User then provides a password to authenticate herself. The SSO protocol builds upon these secure channels to allow the User to authenticate herself to the Service Provider. The SAML SSO Protocol [OAS05] is one such protocol.

However, the use of secure channels is not enough to ensure the security of the application protocol. For example, Google adapted the SAML SSO for use with Google Apps [Goo08]. Unfortunately, this adaptation introduced a flaw, reported in [ACC+08].

The aim of our research programme is to investigate how to analyse such layered protocols. In this paper, we extend the Strand Spaces model [THG98] in order to specify and model layered protocols.

One way to analyse such layered protocols would be to explicitly model both layers; this is the approach taken in [HSN05]. We take the view that it is better to abstract away from the implementation of the Secure Transport Layer and simply to model the services it provides to the Application Layer. This greatly simplifies the analysis of the architecture. Further, such an analysis produces more general results: it allows us to deduce the security of the Application Layer protocol when layered on top of an arbitrary secure transport protocol that provides (at least) the assumed services.

Of course, this approach introduces a proof obligation that the secure transport protocol does indeed provide the services assumed of it. However, such a proof only needs to be done once per transport protocol. An example such proof, for bilateral TLS, appears in [Kam09,KL09]. Such proofs tend to assume that the two layers are independent, so that no message can be replayed from one layer to the other.

Different secure transport protocols will allow or prevent different actions by a dishonest penetrator. Any reasonable transport protocol will allow the penetrator to take part in sessions, to *send* messages using his own identity, and *receive* messages intended for him. Some transport protocols will keep Application Layer messages confidential, such as the transport protocol that encodes Application Layer message m from A to B as

$$TL1_{A,B}(m) = A, B, \{m\}_{PK(B)}$$

(where $PK(B)$ is B's public key). But others will allow the penetrator to *learn* them, such as the transport protocol that encodes m from A to B as

$$TL2_{A,B}(m) = A, B, \{m\}_{SK(A)}$$

[CGRZ03] (where $SK(A)$ is A's secret key). Some transport protocols will allow the penetrator to *fake* messages, causing a regular (i.e. honest) agent to receive an arbitrary Application Layer message known to the penetrator, apparently from some third party; this is the case with encoding *TL1* but not *TL2*. Finally, some transport protocols will allow the penetrator to *hijack* messages, changing either the intended recipient or the apparent sender of the message; for example, with encoding *TL2*, the penetrator may transform the Transport Layer message into $A, C, \{m\}_{SK(A)}$ and redirect it to C; alternatively, with encoding *TL1*, the penetrator may transform the Transport Layer message into $C, B, \{m\}_{PK(B)}$ and re-ascribe it to C.

Our approach is to build an abstract model that describes each of these potential penetrator actions —sending, receiving, learning, faking and hijacking— as a high-level penetrator strand. Of course, the penetrator may also build Application Layer messages himself, pick them apart, or otherwise transform them; we capture these abilities using (slightly adapted versions of) standard penetrator strands.

We assume that different transport protocols deployed in the same system are independent, in the sense that the penetrator cannot directly transform a message sent over one transport protocol into a message over another protocol, other than by performing a receive or learn followed by a send or fake.

In the next section we present the foundations of the model, describing the way we abstractly represent Transport Layer messages and the penetrator's possible actions in high-level bundles. In Section 3 we describe how to specify the properties of secure channels by disallowing appropriate high-level penetrator strands. In Section 4 we prove a normal form lemma that subsequently allows us to restrict our attention to bundles in a particular form. We illustrate our model in Section 5 by using it to analyse a small single sign-on protocol. We sum up and discuss forthcoming work in Section 6.

Related work The work closest to the current paper is [DL08,Dil08]. That paper uses a CSP-style formalism [Ros98], with a view towards analysing protocols using model checking. That paper, like this, defines potential capabilities of the penetrator, and then specifies secure channels by limiting those capabilities. We see the two approaches as complementary: model checking is good for finding attacks; the Strand Spaces approach is good for building the theoretical foundations, and producing proofs of protocols that reveal *why* the protocol is correct.

Armando et al. [ACC07] use LTL to specify security properties of channels, and then use SATMC, a model checker for security protocols, to analyse a fair exchange protocol. In [ACC$^+$08] they analyse SAML SSO and the Google Apps variant using the same techniques. Bella et al. [BLP03] adapt the inductive approach to model authenticated and confidential channels, and use these ideas to verify a certified e-mail delivery protocol. Bugliesi and Focardi [BF08] model secure channels within a variant of the asynchronous pi-calculus. Each of these works captures the properties of secure channels by limiting the penetrator's abilities regarding messages on such channels, although each considers fewer variants of authenticated channels than the current paper.

As noted above, our work is mainly targeted at layered protocol architectures. However, it can also be used to model *empirical channels*, where some messages of a protocol are implemented by human mediation, and so satisfy extra security properties. Creese et al. [CGRZ03] capture such empirical channels, within the context of CSP model checking, again by restricting the penetrator's abilities.

2 The abstract model: High-level bundles

In this section we present high-level bundles, which abstractly model secure transport protocols. We present high-level terms, which capture Transport Layer messages. We then adapt the notion of a strand space [THG98] to such high-level terms. We then describe how we model the penetrator's ability to manipulate both Transport Layer and Application Layer messages. Finally, we define high-level bundles.

2.1 High-level terms and nodes

Let $\mathcal{A}$ be the set of possible messages that can be exchanged between principals in a protocol. The elements of $\mathcal{A}$ are usually referred to as *terms*. As in the

original Strand Spaces model [THG98], $\mathcal{A}$ is freely generated from two disjoint sets, $\mathcal{T}$ (representing tags, texts, nonces, and principals) and $\mathcal{K}$ (representing keys) by means of *concatenation* and *encryption*.

Definition 1 *Compound terms are built by two constructors:*

- $\mathsf{encr} : \mathcal{K} \times \mathcal{A} \to \mathcal{A}$ *representing encryption;*
- $\mathsf{join} : \mathcal{A} \times \mathcal{A} \to \mathcal{A}$ *representing concatenation.*

Conventionally, $\{t\}_k$ is used to indicate that a term t is encrypted with a key k and $t_0 \hat{} t_1$ to denote the concatenation of t_0 and t_1.

The set $\mathcal{K}$ of keys is equipped with a unary injective symmetric operator $\mathsf{inv} : \mathcal{K} \to \mathcal{K}$; $\mathsf{inv}(k)$ is usually denoted k^{-1}. Let $\mathcal{T}_{name} \subseteq \mathcal{T}$ be the set of agent names, ranged over by X, Y; let $\mathcal{T}_{pname} \subseteq \mathcal{T}_{name}$, ranged over by P, be the set of names the penetrator uses when actively participating in a protocol as himself; we let A, B range over names of regular agents.

As in the original Strand Spaces model, a *strand* is a sequence of message transmissions and receptions. A node is the basic element of a strand. Each node n is associated with a message, or high-level term, denoted $msg(n)$. A positive node is used to represent a transmission while a negative node is used to denote reception. Each node communicates over a channel, which may provide some security services to the message; a channel that does not provide any security services is called the *bottom* channel, denoted $\perp$. In our abstract model, messages are modelled as follows.

Definition 2 *Every node n in strand st is associated with a* high-level term *of the form (X, Y, m, c) where:*

- *$m \in \mathcal{A}$ is the Application Layer message.*
- *$X \in \mathcal{T}_{name}$: If n is positive and st is a regular strand, then X is the name of the regular agent who is running st. Otherwise, X refers to the agent that is claimed to have sent m.*
- *$Y \in \mathcal{T}_{name}$: If n is negative and st is a regular strand, then Y is the name of the regular agent who is running st. Otherwise, Y refers to the agent that is intended to receive m.*
- *c is the identifier of the secure channel over which n communicates.*

We write $\hat{\mathcal{A}}$ for the set of high-level terms.

We may use an underscore (_) in the first or second position of the tuple to indicate that the term is not associated with a particular sender or receiver respectively. If n is a regular node then we assume that its term must be associated with a specified sender and receiver.

The following definition is a straightforward adaptation from [THG98]. The relation $n \to n'$ represents inter-strand communication, while $n \Rightarrow n'$ represents flow of control within a strand.

Definition 3 *A directed term is a pair $\langle \sigma, a \rangle$ with $\sigma \in \{+, -\}$ and $a \in \hat{\mathcal{A}}$; we write it as as $+t$ or $-t$. $(\pm\hat{\mathcal{A}})^*$ is the set of finite sequences of directed terms. A typical element of $(\pm\hat{\mathcal{A}})^*$ is denoted by $\langle\langle \sigma_1, a_1 \rangle, ..., \langle \sigma_n, a_n \rangle\rangle$. A strand space over $\hat{\mathcal{A}}$ is a set Σ with a trace mapping $tr : \Sigma \to (\pm\hat{\mathcal{A}})^*$. Fix a strand space Σ.*

1. *A* node *n is a pair $\langle st, i\rangle$, with $st \in \Sigma$ and i an integer satisfying $1 \leq i \leq length(tr(st))$. The set of nodes is denoted by $\mathcal{N}$. We define $msg(n) = tr(st)(i)$. We will say the node n belongs to the strand st.*
2. *There is an edge $n_1 \rightarrow n_2$ if and only if $msg(n_1) = +a$ and $msg(n_2) = -a$ for some $a \in \hat{\mathcal{A}}$. The edge means that node n_1 sends the message a, which is received by n_2, recording a potential causal link between those strands.*
3. *When $n_1 = \langle st, i\rangle$, and $n_2 = \langle st, i+1\rangle$ are members of $\mathcal{N}$, there is an edge $n_1 \Rightarrow n_2$. The edge expresses that n_1 is an immediate causal predecessor of n_2 on the strand st. $n' \Rightarrow^+ n$ is used to denote that n' precedes n (not necessarily immediately) on the same strand.*

We now define the notions of origination and unique origination in the context of a high-level strand space.

Definition 4 *Let Σ be a high-level strand space.*

1. *Let $\mathcal{I}$ be a set of undirected terms. The node $n \in \Sigma$ is an* entry point *for $\mathcal{I}$ iff n is positive and associated with a high-level term (A, B, m, c) for some $m \in \mathcal{I}$, and whenever $n' \Rightarrow^+ n$ and n' is associated with a high-level term (A', B', m', c'), $m' \notin \mathcal{I}$.*
2. *An undirected term t* originates *on a node n iff n is an entry point for the set of messages that contain t as a subterm.*
3. *An undirected term t is* uniquely originating *in a set of nodes $S \subset \mathcal{N}$ iff there is a unique $n \in S$ such that t originates on n.*

2.2 The penetrator

We can classify the activities of the penetrator, according to their effects on Application Layer messages:

- Actions that are used to construct or pick apart Application Layer messages;
- Actions that are used to handle high-level terms, affecting the Transport Layer "packaging" without modifying the corresponding Application Layer messages.

The first type of actions is used to transform and create messages of the form $(\_, \_, m, \bot)$, i.e. messages that are sent on the bottom channel without being associated with a particular sender or receiver. To model them, we adapt the standard penetrator strands from [THG98] to handle high-level terms.

Definition 5 *A standard penetrator strand in a high level bundle is one of the following:*

M. *Text message: $\langle +(\_, \_, r, \bot)\rangle$ where $r \in \mathcal{T}_P$;*
K. *Key: $\langle +(\_, \_, k, \bot)\rangle$ where $k \in \mathcal{K}_\mathcal{P}$;*
C. *Concatenation: $\langle -(\_, \_, t_0, \bot), -(\_, \_, t_1, \bot), +(\_, \_, t_0 \hat{} t_1, \bot)\rangle$;*
S. *Separation into components: $\langle -(\_, \_, t_0 \hat{} t_1, \bot), +(\_, \_, t_0, \bot), +(\_, \_, t_1, \bot)\rangle$;*
E. *Encryption: $\langle -(\_, \_, k, \bot), -(\_, \_, t, \bot), +(\_, \_, \{t\}_k, \bot)\rangle$ where $k \in \mathcal{K}$;*

D. *Decryption:* $\langle -(\_, \_, k^{-1}, \bot), -(\_, \_, \{t\}_k, \bot), +(\_, \_, t, \bot)\rangle$ *where* $k \in \mathcal{K}$.

The second type of penetrator actions only affects the "packaging" of the Application Layer message, i.e. it only affects the first, second, and fourth components of a high-level term. These paths are used to perform the following activities:

1. **Send**: the penetrator may send an Application Layer message m by creating a Transport Layer message with payload m, and inserting it in the network using a penetrator's identity.
2. **Receive**: the penetrator may receive an Application Layer message m as a payload of a Transport Layer message that was sent for him by a regular agent.
3. **Learn**: the penetrator may intercept and learn an Application Layer message m from a Transport Layer message with a payload m that was exchanged between regular agents.
4. **Fake**: the penetrator may fake an Application Layer message by creating a Transport Layer message with payload m, and inserting it in the network dishonestly (i.e. using another agent's identity).
5. **Hijack**: the penetrator may change the sender and/or receiver field in a previously sent Transport Layer message without changing the Application Layer message; the penetrator can perform hijacking in three ways [DL08]:
   (a) **Re-ascribe**: the penetrator may *re-ascribe* a previously sent message by intercepting and sending it using another agent's identity.
   (b) **Redirect**: the penetrator may *redirect* a previously sent message by intercepting it and sending it to a different agent.
   (c) **Re-ascribe/redirect**: the penetrator may re-ascribe and redirect a previously sent message at the same time.

We abstractly model each of the penetrator paths defined above as a high-level penetrator strand that sends and receives terms in the form (A, B, m, c).

Definition 6 *A penetrator strand in a high-level bundle is either a standard penetrator strand or a high-level penetrator strand of one of the following forms:*

SD. *Sending:* $\langle -(\_, \_, m, \bot), +(P, B, m, c)\rangle$ *where* $P \in \mathcal{T}_{pname}$ *and* $B \notin \mathcal{T}_{pname}$*;*
RV. *Receiving:* $\langle -(A, P, m, c), +(\_, \_, m, \bot)\rangle$ *where* $P \in \mathcal{T}_{pname}$ *and* $A \notin \mathcal{T}_{pname}$*;*
LN. *Learning:* $\langle -(A, B, m, c), +(\_, \_, m, \bot)\rangle$ *where* $A, B \notin \mathcal{T}_{pname}$*;*
FK. *Faking:* $\langle -(\_, \_, m, \bot), +(A, B, m, c)\rangle$ *where* $A, B \notin \mathcal{T}_{pname}$*;*
HJ. *Hijacking:* $\langle -(X, Y, m, c), +(X', Y', m, c)\rangle$ *such that* $X \neq X'$ *or* $Y \neq Y'$.

As an example, Figure 1 illustrates part of a bundle, where the penetrator uses several different strands to transform the high level message $(S, P, A\hat{}N, c)$ into $(S, B, P\hat{}N, c)$.

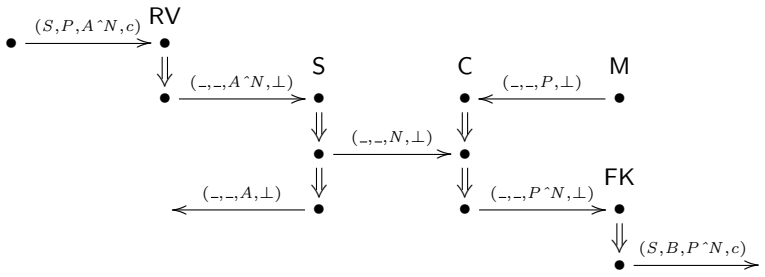


Fig. 1. Transforming a high-level term.

2.3 High-Level Bundles

A *high-level bundle* is a finite subgraph of $\langle \mathcal{N}, (\rightarrow \cup \Rightarrow) \rangle$ for which the edges express the causal dependencies of the nodes.

Definition 7 *[THG98] Suppose $\rightarrow_{\mathcal{B}} \subset \rightarrow$, $\Rightarrow_{\mathcal{B}} \subset \Rightarrow$, and $\mathcal{B} = \langle \mathcal{N}_{\mathcal{B}}, \rightarrow_{\mathcal{B}} \cup \Rightarrow_{\mathcal{B}} \rangle$ is a subgraph of $\langle \mathcal{N}, \rightarrow \cup \Rightarrow \rangle$. $\mathcal{B}$ is a bundle if (1) $\mathcal{N}_{\mathcal{B}}$ and $\rightarrow_{\mathcal{B}} \cup \Rightarrow_{\mathcal{B}}$ are finite; (2) If $n_2 \in \mathcal{N}_{\mathcal{B}}$ and $msg(n_2)$ is negative, then there is a unique n_1 such that $n_1 \rightarrow_{\mathcal{B}} n_2$; (3) If $n_2 \in \mathcal{N}_{\mathcal{B}}$ and $n_1 \Rightarrow n_2$ then $n_1 \Rightarrow_{\mathcal{B}} n_2$; and (4) $\mathcal{B}$ is acyclic. We write $\preceq_{\mathcal{B}}$ for $(\rightarrow_{\mathcal{B}} \cup \Rightarrow_{\mathcal{B}})^*$.*

The relation $\preceq_{\mathcal{B}}$ expresses the causal relationship in the high-level bundle $\mathcal{B}$.

Proposition 8 *Let $\mathcal{B}$ be a high-level bundle. Then $\preceq_{\mathcal{B}}$ is a partial order, i.e. a reflexive, antisymmetric, transitive relation. Every non-empty subset of the nodes in $\mathcal{B}$ has a $\preceq_{\mathcal{B}}$-minimal member.*

Definition 9 *Bundles $\mathcal{B}$ and $\mathcal{B}'$ in a strand space Σ are* equivalent *iff they have the same regular strands.*

3 Modelling Secure Channels

So far we allow high-level bundles with *arbitrary* penetrator strands. In this section we restrict these strands to capture properties of secure channels. Our approach follows [DL08]. We begin with confidential channels, and then provide the building blocks of authenticated channels. Each channel is associated with a specification that states which of these properties it satisfies.

Confidential channels protect the confidentiality of the messages sent on them. If message (A, B, m, c) is sent over a confidential channel c, the penetrator cannot deduce m if the message was not intended for him. However, he can still see the high-level message. We can define a secure channel c to satisfy the confidentiality property C in terms of the penetrator's activity as follows.

Definition 10 (Confidential Channels) *Let channel c satisfy C. Then there is no* LN *strand of the form $\langle -(A, B, m, c), +(\_, \_, m, \bot) \rangle$ where $A, B \notin \mathcal{T}_{pname}$ in any high-level bundle.*

For example, if the Transport Layer protocol encodes the high-level term (A, B, m, c) as $A\hat{}\{m\}_{PK(B)}$, where $PK(B)$ is B's public key, then it provides a confidential channel. We write $C(c)$ to indicate that c satisfies C, and similarly for the properties we define below.

If a channel is non-fakable, then the penetrator cannot create and send an application message using another agent's identity.

Definition 11 (No faking) *Let channel c satisfy NF. Then there is no* FK *strand of the form $\langle -(\_, \_, m, \bot), +(A, B, m, c)\rangle$ where $A, B \notin \mathcal{T}_{pname}$ in any high-level bundle.*

For example, if the Transport Layer protocol encodes the high-level term (A, B, m, c) as $B\hat{}\{m\}_{SK(A)}$, where $SK(A)$ is A's secret key, then it provides a non-fakable channel.

We now consider various restrictions on hijacking. In each case we do not want to prevent HJ strands of the form $\langle -(X, Y, m, c), +(X', Y', m, c)\rangle$ where (i) if $C(c)$ then $Y \in \mathcal{T}_{pname}$, and (ii) if $NF(c)$ then $X' \in \mathcal{T}_{pname}$: in such cases the penetrator could learn m (via a RV or LN strand) and then produce $+(X', Y', m, c)$ (via a SD or FK strand) to produce the same effect.

If a channel is non-re-ascribable, then the penetrator cannot intercept a previously sent message and send it using a different sender's identity. Following [DL08], we distinguish between two notions of no re-ascribing:

- *No re-ascribing* where the penetrator cannot re-ascribe messages using any identity;
- *No honest re-ascribing* where the penetrator cannot re-ascribe messages using an honest identity, but can still re-ascribe messages using a penetrator's identity.

For example, if the Transport Layer protocol encodes the high-level term (A, B, m, c) as $\{\{m\}_{PK(B)}\}_{SK(A)}$, then the penetrator P may replace the signature using $SK(A)$ with his own signature using $SK(P)$, so as to re-ascribe the message to himself; however, he cannot re-ascribe the message to an honest agent. On the other hand, if (A, B, m, c) is encoded as $\{\{m, A\}_{PK(B)}\}_{SK(A)}$, then he can no longer re-ascribe the message to himself.

We define *non re-ascribable* channels as follows.[1]

Definition 12 (No honest re-ascribing) *Let channel c satisfy NRA^-. Then for every* HJ *strand of the form $\langle -(X, Y, m, c), +(X', Y', m, c)\rangle$ in a high-level bundle, one of the following holds: (a) $X = X'$, i.e. no re-ascribing takes place; (b) $X' \in \mathcal{T}_{pname}$, i.e. the message is re-ascribed with a penetrator's identity; or (c) if $C(c)$ then $Y \in \mathcal{T}_{pname}$, and if $NF(c)$ then $X' \in \mathcal{T}_{pname}$, i.e., as discussed above, the penetrator can learn the underlying Application Layer message and then send or fake the message.*

[1] Some of the details of these definitions are a little delicate, and are necessary for some of the future work discussed in Section 6.

Definition 13 (No re-ascribing) *Let channel c satisfy NRA. Then for every* HJ *strand of the form $\langle -(X, Y, m, c), +(X', Y', m, c)\rangle$ in a high-level bundle, one of the following holds: (a) $X = X'$, i.e. no re-ascribing takes place; (b) $X, X' \in \mathcal{T}_{pname}$, i.e. the message is re-ascribed from one penetrator identity to another; or (c) if $C(c)$ then $Y \in \mathcal{T}_{pname}$, and if $NF(c)$ then $X' \in \mathcal{T}_{pname}$.*

If a channel is non-redirectable, the penetrator cannot intercept a previously sent message and send it for a different receiver. As with re-ascribing, we distinguish between two notions of no redirecting:

- *No-redirecting* where the penetrator cannot redirect any message;
- *No-honest redirecting* where the penetrator cannot redirect messages sent to honest participants, but can redirect messages sent to himself.

For example, if the Transport Layer protocol encodes the high-level term (A, Y, m, c) as $\{\{m\}_{SK(A)}\}_{PK(Y)}$, then the penetrator P can transform a message for himself, i.e. $\{\{m\}_{SK(A)}\}_{PK(P)}$, into one for B, i.e. $\{\{m\}_{SK(A)}\}_{PK(B)}$, and so redirect it to B; however he cannot redirect a message sent to an honest agent. On the other hand, if (A, Y, m, c) is encoded as $\{\{m, Y\}_{SK(A))}\}_{PK(Y)}$, then he can no longer redirect a message sent to himself.

We define non-redirectable channels as follows.

Definition 14 (No honest redirecting) *Let channel c satisfy NRD^-. Then for every* HJ *strand of the form $\langle -(X, Y, m, c), +(X', Y', m, c)\rangle$ in a high-level bundle, one of the following holds: (a) $Y = Y'$, i.e. no redirecting takes place; (b) $Y \in \mathcal{T}_{pname}$, i.e. the original message was sent to the penetrator; or (c) if $C(c)$ then $Y \in \mathcal{T}_{pname}$, and if $NF(c)$ then $X' \in \mathcal{T}_{pname}$.*

Definition 15 (No-redirecting) *Let channel c satisfy NRD. Then for every* HJ *strand of the form $\langle -(X, Y, m, c), +(X', Y', m, c)\rangle$ in a high-level bundle, one of the following holds: (a) $Y = Y'$, i.e. no redirecting takes place; (2) $Y, Y' \in \mathcal{T}_{pname}$, i.e. the message is redirected from one penetrator identity to another; or (c) if $C(c)$ then $Y \in \mathcal{T}_{pname}$, and if $NF(c)$ then $X' \in \mathcal{T}_{pname}$.*

4 Normal and abstractly efficient bundles

Bundles can contain various types of *redundancy*. For example, an encryption edge immediately followed by a decryption edge just reproduces the original term: this redundancy can be removed to produce an equivalent bundle. It is clearly simpler if we can restrict our attention to bundles without such redundancies. This is the question we consider in this section.

Definition 16 *In a high-level bundle, a $\Rightarrow^+$ edge is* constructive *if it is part of an* E, C, SD *or* FK *strand. It is* destructive *if it is part of a* D, S, LN *or* RV *strand. An edge is* non-destructive *if it is constructive or part of an* HJ *strand. Similarly, an edge is* non-constructive *if it is destructive or part of an* HJ *strand.*

Definition 17 *A high-level bundle $\mathcal{B}$ is* normal *iff for any penetrator path of $\mathcal{B}$, no non-destructive edge precedes a non-constructive edge.*

Proposition 18 *For every high-level bundle $\mathcal{B}$, there exists an equivalent high-level normal bundle $\mathcal{B}'$. Moreover, the penetrator nodes of $\mathcal{B}'$ form a subset of the penetrator nodes of $\mathcal{B}$ and the ordering $\preceq_{\mathcal{B}'}$ is a restriction of the ordering $\preceq_{\mathcal{B}}$.*

Proof. (Sketch.) The proof proceeds by showing that whenever a non-destructive edge precedes a non-constructive edge, an equivalent bundle can be found without this redundancy. For standard penetrator strands, the proof is as in [GT01]. Figure 2 (a)–(d) gives some examples of how to replace redundancies arising from high-level penetrator strands. A simple case analysis shows no redundancy involves a standard penetrator strand and a high-level penetrator strand.

Normal bundles may still contain redundancies. For example, an LN strand followed by an SD strand may be replaced by an HJ strand (or a transmission edge if the identities match).

Definition 19 *A high-level bundle $\mathcal{B}$ is* abstractly efficient *if every penetrator path p that starts at n_1 such that $msg(n_1) = +(X, Y, m, c)$, and ends at n_2 such that $msg(n_2) = -(X', Y', m, c)$, consists of a single* HJ *strand or else there is a transmission edge between n_1 and n_2.*

Proposition 20 *For every high-level bundle, there exists an equivalent high-level efficient bundle that is also normal.*

Proof. (Sketch.) Figure 2 (e)–(f) gives examples of how some of the remaining redundancies can be removed.

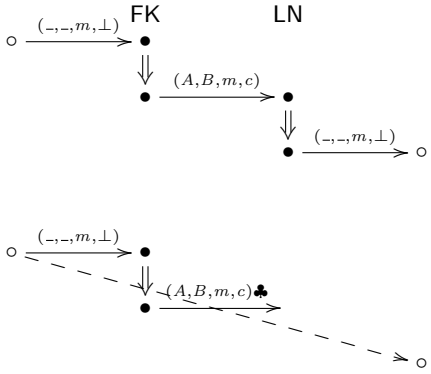
5 Example: a single sign-on protocol

In this example we illustrate our definitions of high-level bundles and secure channels via a small example. We consider a single sign-on protocol that authenticates a User U to a Service Provider SP, with the help of an Identity Provider IdP.

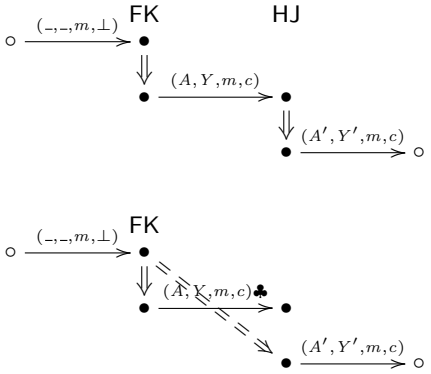
We will use a secure channel c that satisfies $C \wedge NRD^{-}$. It is reasonable to suppose that the Service Provider and Identity Provider each has a public key certificate, so unilateral TLS can be used to establish an authenticated channel to them; we believe that this channel satisfies $C \wedge NRD^{-}$. For messages sent from the Identity Provider to the User, the channel could be implemented using unilateral TLS to authenticate the Identity Provider, combined with the User sending a password to authenticate herself; we believe that this channel satisfies $C \wedge NF \wedge NRD \wedge NRA$, which is stronger than is required.

We will consider the following protocol, where $\rightarrow_c$ indicates messages sent using channel c, and $\rightarrow$ indicate messages sent on the bottom channel.

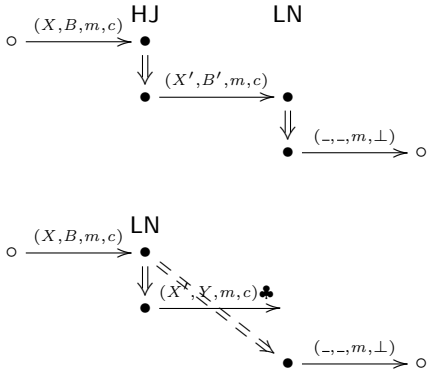
$$
\begin{array}{lrcl:l}
0. & U & \rightarrow & SP & U \mathbin{\hat{}} IdP \\
1. & SP & \rightarrow_c & IdP & 1 \mathbin{\hat{}} SP \mathbin{\hat{}} U \mathbin{\hat{}} N \\
2. & IdP & \rightarrow_c & U & 2 \mathbin{\hat{}} IdP \mathbin{\hat{}} SP \mathbin{\hat{}} N \\
3. & U & \rightarrow_c & SP & 3 \mathbin{\hat{}} U \mathbin{\hat{}} IdP \mathbin{\hat{}} N
\end{array}
$$



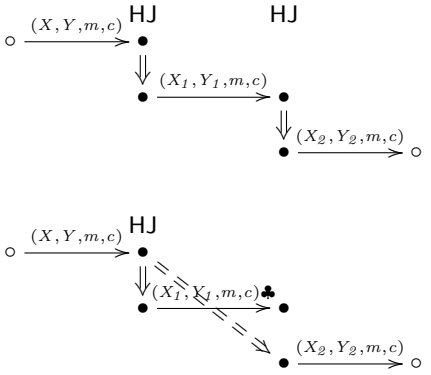
(a) A path containing an FK-LN redundancy, replaced by a transmission edge.



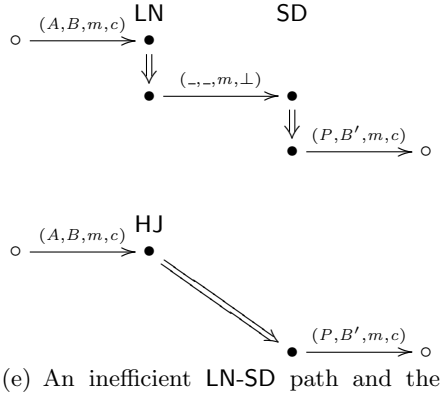
(b) A path containing an FK-HJ redundancy, replaced by an FK strand.



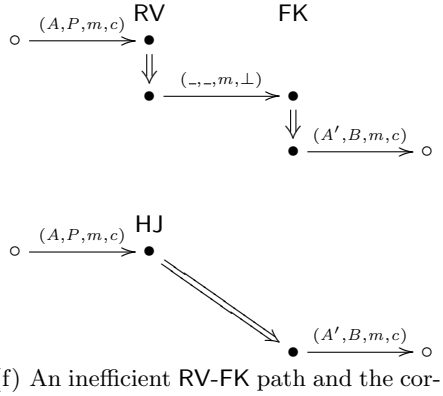
(c) A path containing an HJ-LN redundancy, replaced by an LN strand.



(d) A path containing an HJ-HJ redundancy, replaced by an HJ strand.



(e) An inefficient LN-SD path and the corresponding efficient path.



(f) An inefficient RV-FK path and the corresponding efficient path.

Fig. 2. Redundancies and how to eliminate them. ♣ indicates a discarded message.

Here N is a fresh unpredictable value; "*1*", "*2*" and "*3*" are distinct tags used to ensure unique readability of the messages. Message 0 is sent across the bottom channel to initiate the protocol. SP then creates a fresh nonce which is passed via IdP to U, and then back to U in order to authenticate U to SP.

In order to model this protocol, we start by defining regular strands for each of the three roles.

- Strands of the form $User(U, SP, IdP, N)$ have trace

$$\langle + \ (U,\ SP,\ \ U\,\hat{}\,IdP,\ \ \bot),\\ -\ (IdP,\ U,\ 2\,\hat{}\,IdP\,\hat{}\,SP\,\hat{}\,N,\ c),\\ +\ (U,\ SP,\ 3\,\hat{}\,U\,\hat{}\,IdP\,\hat{}\,N,\ c)\,\rangle.$$

- Strands of the form $ServProv(SP, U, IdP, N)$ have trace

$$\langle - \ (U,\ SP,\ \ U\,\hat{}\,IdP,\ \ \bot),\\ +\ (SP,\ IdP,\ 1\,\hat{}\,SP\,\hat{}\,U\,\hat{}\,N,\ c),\\ -\ (U,\ SP,\ 3\,\hat{}\,U\,\hat{}\,IdP\,\hat{}\,N,\ c)\,\rangle.$$

- Strands of the form $IdProv(IdP, U, SP, N)$ have trace

$$\langle - \ (SP,\ IdP,\ 1\,\hat{}\,SP\,\hat{}\,U\,\hat{}\,N,\ c),\\ +\ (IdP,\ U,\ 2\,\hat{}\,IdP\,\hat{}\,SP\,\hat{}\,N,\ c)\,\rangle.$$

We will therefore consider bundles $\mathcal{B}$ containing strands of the above form (and no other regular strands). Further, since each nonce N is freshly generated, we assume that for every $ServProv(SP, U, IdP, N)$ strand st containing at least two nodes in the bundle, N originates uniquely at $(st, 2)$.

Consider a bundle $\mathcal{B}$ containing all three nodes of a Service Provider strand $st = ServProv(SP, U, IdP, N)$, and such that $SP, U, IdP \notin \mathcal{T}_{pname}$. We aim to show that there is a corresponding $User$ strand in $\mathcal{B}$, i.e., the User is authenticated to the Service Provider. By Proposition 20, we may, without loss of generality, assume that $\mathcal{B}$ is normal and abstractly efficient.

The reason the protocol works is that only SP, U and IdP can obtain N. The lemma below captures this. Let X be the set containing the terms $(SP,\ IdP,\ 1\,\hat{}\,SP\,\hat{}\,U\,\hat{}\,N,\ c)$, $(IdP,\ U,\ 2\,\hat{}\,IdP\,\hat{}\,SP\,\hat{}\,N,\ c)$, and $(U,\ SP,\ 3\,\hat{}\,U\,\hat{}\,IdP\,\hat{}\,N,\ c)$.

Lemma 21 *Every occurrence of N on a regular node is within a high-level term from X; N does not occur within any high-level term of the form $(\_, \_, m, \bot)$.*

Proof. Suppose for a contradiction that the result does not hold. Let n be a $\preceq_{\mathcal{B}}$-minimal node where this occurs (there must be a minimal such node by Proposition 8). Clearly $n \neq (st, 2)$, since $msg(st, 2)$ is in X. Since N originates uniquely at $(st, 2)$, N does not originate at n. Hence one of the following holds.

- n is a positive regular node. Then there must be some node n' such that $n' \Rightarrow^+ n$ and N occurs within $msg(n')$. Then, by the assumed $\preceq_{\mathcal{B}}$-minimality of n, $msg(n') \in X$; hence the strand containing n and n' transforms a message from X into a message not in X. But no regular strand can do this; for example:

- If an Identity Provider strand receives a message from X, it is necessarily of the form (SP, IdP, $1\hat{}SP\hat{}U\hat{}N$, c); it will then send (IdP, U, $2\hat{}IdP\hat{}SP\hat{}N$, c), which is also in X: the presence of the SP and U fields within message 1 is important here.
- If a User strand receives a message from X, it is necessarily of the form (IdP, U, $2\hat{}IdP\hat{}SP\hat{}N$, c); it will then send (U, SP, $3\hat{}U\hat{}IdP\hat{}N$, c), which is also in X: the presence of the SP field within message 2 is important here.

– n is either a negative regular node containing N outside X, or a penetrator node containing N in a term of the form $(\_, \_, m, \bot)$. In each case, the term is produced by a penetrator path that starts at a regular node n' containing a term from X. Since $\mathcal{B}$ is normal and abstractly efficient, every such penetrator path must start with an RV or LN strand, or comprise a single HJ strand. Clearly no RV strand can operate on terms from X. Since c is a confidential channel, no LN strand can operate on messages from X. Since c satisfies $C \wedge NRD^{-}$, every HJ strand either: (case (a) of Definition 14) changes only the first field of high-level messages, but no honest strand will accept the result of transforming a term from X in this way; or (cases (b) and (c) of Definition 14) operates on high-level messages whose second field is an element of $\mathcal{T}_{pname}$, so cannot operate on messages from X. □

Now consider the term (U, SP, $3\hat{}U\hat{}IdP\hat{}N$, c) received at $(st, 3)$. From the above lemma, the term could not have been produced by an FK strand. Using this and the fact that the bundle is normal and abstractly efficient, the term must result from either a transmission edge, or an HJ strand, from a term from X. The latter case cannot occur (since each HJ strand changes either the sender or receiver field). An analysis of the honest strands then shows that the message is transmitted from the final node of a $User(U, SP, IdP, N)$ strand.

A similar analysis could be used to show the presence of a corresponding Identity Provider strand; we omit the details.

It is possible to simplify the protocol slightly, by removing some fields. However, we do need the fields U and SP in message 1, and SP in message 2 to ensure that Lemma 21 is satisfied, in particular by honest Identity Provider and User strands. Further (but arguably less importantly), the IdP field in message 3 is needed to ensure the User and Service Provider agree upon the identity of the Identity Provider. Finally, the presence of the U field in message 3 simplifies the proof slightly.

6 Conclusion

In this paper we have described how to model secure channels within the Strand Spaces formalism. We represent messages sent over the network using high-level terms, which abstract away from the implementation of the secure transport protocol. We then abstractly modelled ways in which the penetrator can operate upon such high level terms: to obtain the underlying Application Layer message

(either honestly or dishonestly); to have an honest agent receive the Application Layer message (either apparently from the penetrator or some third party); or to hijack the message, to change either the recipient or the apparent sender. We specified properties of secure channels by restricting the capabilities available to the penetrator. Finally, we illustrated the model by using it to verify a property of a simple single sign-on protocol: we believe that the proof helps to explain *why* the protocol is correct.

This is the first of a planned series of papers reporting work from [Kam09]. We briefly discuss some of the results here.

Many secure transport protocols group messages together into *sessions*, so that the recipient of messages receives an assurance that the sender sent those messages as part of the same session. For example, a single sign-on protocol is normally used as a prelude to some session: the Service Provider wants to be sure that all the messages in that session came from the same User who was authenticated by the single sign-on. Further, some transport protocols give the recipient a guarantee that the messages were received in the same order in which they were sent. These properties are captured in [Kam09, Chapter 5].

In this paper we have presented high-level bundles, which abstract away from the implementation of the secure transport protocol. As mentioned in the Introduction, one could also model layered architectures by explicitly modelling the transport protocol, in low-level bundles. In [Kam09, Chapter 6] the relationship between these models is described, and it is shown that —subject to certain independence assumptions— for every low-level bundle, there is a high-level bundle that abstracts it. Hence the abstraction is sound: by verifying a protocol in a high-level Strand Space, one can deduce that the implementation of the protocol, as modelled in the low-level Strand Space, is also correct.

In [DL08] it is shown that not all combinations of the properties from Section 3 are distinct, and a hierarchy of different properties is —informally— derived. In [Kam09, Chapter 7] the same result is —more formally— obtained for our Strand Spaces definitions.

In Section 5, we performed a direct verification of the example protocol. In [Kam09, Chapter 7], a number of verification-oriented rules are presented. Some rules concern when an honest agent receives a message over a particular secure channel, and allow one to deduce facts about how that message was produced. Further rules allow one to verify the secrecy of certain terms, while others adapt the Authentication Tests of [GT00] to high-level bundles. In [Kam09, Chapter 8], these rules are used in a number of examples, concerning both layered protocol architectures and empirical channels.

References

[ACC07] A. Armando, R. Carbone, and L. Compagna. LTL model checking for security protocols. In *20th IEEE Computer Security Foundations Symposium*, 2007.

[ACC+08] A. Armando, R. Carbone, L. Compagna, J. Cuellar, and L. Tobarra. Formal analysis of SAML 2.0 web browser single sign-on: Breaking the SAML-based

single sign-on for Google Apps. In *The 6th ACM Workshop on Formal Methods in Security Engineering (FMSE 2008)*, 2008.

[BF08] M. Bugliesi and R. Focardi. Language based secure communication. In *Proceedings of the 21st IEEE Computer Security Foundations Symposium*, 2008.

[BLP03] G. Bella, C. Longo, and L. Paulson. Verifying second-level security protocols. In *Theorem Proving in Higher Order Logics: 16th International Conference, (TPHOLs 2003)*, 2003.

[CGRZ03] S.J. Creese, M.H. Goldsmith, A. W. Roscoe, and I. Zakiuddin. The attacker in ubiquitous computing environments: formalising the threat model. In *Proceedings of the 1st International Workshop on Formal Aspects in Security and Trust (FAST)*, 2003.

[Dil08] Christopher Dilloway. *On the Specification and Analysis of Secure Transport Layers*. DPhil thesis, Oxford University, 2008.

[DL08] Christopher Dilloway and Gavin Lowe. Specifying secure transport layers. In *21st IEEE Computer Security Foundations Symposium (CSF 21)*, 2008.

[Goo08] Google. Web-based reference implementation of SAML-based SSO for Google Apps. `http://code.google.com/apis/apps/sso/saml_reference_implementation_web.html`, 2008.

[GT00] Joshua D. Guttman and F. Javier Thayer. Authentication tests. In *IEEE Symposium on Security and Privacy*, pages 96–109, 2000.

[GT01] Joshua D. Guttman and F. Javier Thayer. Authentication tests and the structure of bundles. *Theoretical Computer Science*, 2001.

[HSN05] S. M. Hansen, J. Skriver, and H. R. Nielson. Using static analysis to validate the SAML single sign-on protocol. In *Proceedings of the 2005 Workshop on Issues in the Theory of Security (WITS '05)*, 2005.

[Kam09] Allaa Kamil. *The Modelling and Analysis of Layered Security Architectures in Strand Spaces*. DPhil thesis, Oxford University, 2009. Forthcoming.

[KL09] Allaa Kamil and Gavin Lowe. Analysing TLS in the Strand Spaces model. Submitted for publication, 2009.

[OAS05] OASIS Security Services Technical Committee. Security assertion markup language (SAML) v2.0 technical overview, 2005. Available from `http://www.oasis-open.org/committees/security/`.

[Ros98] A. W. Roscoe. *The Theory and Practice of Concurrency*. Prentice Hall, 1998.

[THG98] F. Javier Thayer, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Why is a security protocol correct?. In *IEEE Symposium on Research in Security and Privacy*, pages 160–171. IEEE Computer Society Press, 1998.

[Tho00] Stephen Thomas. *SSL and TLS: Securing the Web*. Wiley, 2000.