Analysing Applications Layered on Unilaterally Authenticating Protocols

Thomas Gibson-Robinson and Gavin Lowe

Department of Computer Science, University of Oxford
Wolfson Building, Parks Road, Oxford, OX1 3QD, UK
{thomas.gibson-robinson, gavin.lowe}@cs.ox.ac.uk

Abstract. There are many approaches to proving the correctness of application-layer protocols that are layered on secure transport protocols, such as TLS. One popular approach is *verification by abstraction*, in which the correctness of the application-layer protocol is proven under the assumption that the transport layer satisfies certain properties. Following this approach, we adapt the strand spaces model in order to analyse application-layer protocols that depend on *unilaterally authenticating secure transport protocols*, such as unilateral TLS. We develop proof rules that enable us to prove the correctness of application-layer protocols that use either unilateral or bilateral secure transport protocols, and illustrate them by proving the correctness of WebAuth, a single-sign-on protocol that makes extensive use of unilateral TLS.

1 Introduction

Many application-layer protocols make use of secure transport protocols, such as TLS [1], to provide security guarantees, such as confidentiality and authentication. There are two different methods of verifying such application-layer protocols: either a combined protocol formed from the composition of the application protocol and the underlying secure transport protocol is derived, or the application-layer protocol is analysed assuming the transport protocol satisfies certain properties (i.e. *verification by abstraction*). In this paper we concentrate on using the second method as it offers a number of advantages. Proofs of correctness using this method are considerably simpler (as the combined protocols are often very large); further, since the proof is not dependent on a particular secure transport protocol, any secure transport protocol that provides the same properties can be used instead.

Most existing work [2,3,4,5,6,7] on verifying application-layer protocols by abstraction has focused on modelling the security guarantees provided by *bilateral* secure transport protocols, such as bilateral TLS, where each participant is authenticated to the other. However, many application-layer protocols, in particular those used on the web, are unable to make use of bilateral TLS, because each party must possess a public-key certificate. Therefore, web-based protocols almost exclusively make use of *unilateral* TLS. This alters the security guarantees to the server, as the client is no longer authenticated. However, the server can assume that all messages came from the same source, and that any messages the server sends will be received only by the source; further, the client's guarantees are unchanged. Despite its limitations, this means that unilateral TLS provides sufficient guarantees for many application-layer protocols.

If we wish to verify (by abstraction) application-layer protocols that make use of unilateral TLS, or other unilaterally authenticating secure transport protocols, we need to be able to formalise the security guarantees that the transport protocol provides. In this paper we investigate techniques for proving the correctness of such application-layer protocols. We build on the *high-level strand spaces model* [8,9], an extension of the original strand spaces model [10].

Whilst the high-level strand spaces model is able to model the security guarantees provided by bilateral TLS accurately, it is unable to model those given by unilateral TLS. Unilaterally authenticating secure transport protocols differ from standard secure transport protocols in that the server cannot deduce anything about the identity of the client. Therefore, in this paper we enhance the model in order to capture precisely the authentication guarantees provided by unilaterally authenticating secure transport protocols.

These changes turned out to be less straightforward than we had anticipated due to how the session properties —which provide guarantees to a recipient that several messages were sent within the same session— were formalised. In particular, the session properties were previously defined by looking at the overall behaviour in a bundle which also made proofs difficult to construct and understand. Further, it did not permit multiple sessions between the same participants on the same strand; we modify the model to allow multiple sessions per strand. We present the new model in Section 2.

In Section 3 we develop some general proof rules that can be used for proving the correctness of application-layer protocols that use either unilateral or bilateral secure transport protocols. We then show the usefulness of these proof rules and the power of the new model by proving the correctness of a single-sign-on protocol, WebAuth [11]. In Section 4 we give an informal overview of WebAuth before formalising it in a strand space definition. In Section 5 we prove the correctness of WebAuth by proving three propositions that show what each principle can infer having completed a run; our analysis reveals a subtlety concerning the strength of authentication guarantees to the application server, and a requirement on the user that may not be obvious to all users. We sum up in Section 6.

Throughout this paper we do not consider issues such as the penetrator reordering messages within of a session, session corruptions or perfect forward secrecy. We see no issue with supporting them in a more complex model, should an application require it.

2 The High-Level Strand Spaces Model

We start by presenting the basics of the high-level strand spaces model.

Definition 1. $\mathcal{T}_{names}$ denotes the set of principals' names; it has two subsets: the regular names $\mathcal{T}_{names}^{reg}$ and the penetrator names $\mathcal{T}_{names}^{pen}$. We further assume a value ?; we use this in our model when no name is authenticated (see below); we have $\mathcal{T}^{reg}_{names} \cap \mathcal{T}^{pen}_{names} = \{?\}$. The set $\mathcal{K}$ of cryptographic keys has a distinct subset $\mathcal{K}_{sym}$ of symmetric keys; the inverse of a key $K \in \mathcal{K}$ is denoted K^{-1}. The set $\mathcal{T}$ of atoms contains atomic messages, and includes $\mathcal{T}_{names} \cup \mathcal{K}$.

The set $\mathcal{A}$ of *terms* is defined as the closure of $\mathcal{T}$ under encryption (written as $\{m\}_K$) and concatenation (written $t_1 \hat{}\, t_2$). We assume an ordering relation $\sqsubseteq$ over $\mathcal{A}$, such that $t_1 \sqsubseteq t_2$ iff t_1 can be obtained from t_2 by a finite number of decryption and splitting operations.

The set $\mathcal{A}_\mathcal{P} \subseteq \mathcal{A}$ of public terms is the set of terms that the penetrator initially knows. The sets $\mathcal{T}_\mathcal{P} = \mathcal{A}_\mathcal{P} \cap \mathcal{T}$ and $\mathcal{K}_\mathcal{P} = \mathcal{A}_\mathcal{P} \cap \mathcal{K}$ are the sets of atoms and keys initially known to the penetrator, respectively.

We now consider how to model application-layer messages. We consider a *channel* as an object that allows two participants to exchange messages: messages sent at one end of the channel are intended to be received at the other end. A fundamental property of TLS is that a principal is able to send or receive on a channel only if she has the relevant cryptographic keys; these are different in different channels, which prevents messages being replayed between sessions. A *channel end* encapsulates all such information that is required to communicate on a channel; as we abstract away from the details of the transport protocol, we treat channel ends as opaque values. For generality, we also consider channels (for protocols that are weaker than TLS) that do not provide such a separation between sessions; we denote the channel end by ? in such cases. Messages will be addressed by *channel endpoints* which consist of a name and a channel end.

Definition 2. We assume a set of *channel types*, *Channels*, that contains a value $\perp$ that represents the channel that provides no security guarantees. We write $\textsc{tls}^{c \to s}$ and $\textsc{tls}^{s \to c}$ to represent the channel types of a unilateral TLS connection from client to server and server to client respectively.

$\mathcal{C}$ denotes the set of *channel ends*. It has two subsets: *penetrator channel ends*, $\mathcal{C}^{pen}$, known to the penetrator; and *regular channel ends*, $\mathcal{C}^{reg}$, known only to regular agents; we have $\mathcal{C}^{reg} \cap \mathcal{C}^{pen} = \{?\}$.

The set of *endpoints* $\mathcal{I}$ is defined as $\mathcal{T}_{names} \times \mathcal{C}$; the set of regular endpoints $\mathcal{I}^{reg}$ as $\mathcal{T}^{reg}_{names} \times \mathcal{C}^{reg}$; and the set of penetrator endpoints $\mathcal{I}^{pen}$ as $\mathcal{T}^{pen}_{names} \times \mathcal{C}^{pen}$. We denote a typical member, $(A, \psi) \in \mathcal{I}$, as A_ψ.

High-level terms model data being sent across the network. They are of the form $(A_\psi,\ B_\phi,\ m,\ c)$, representing an application-layer term m being sent from A's channel end ψ to B's channel end ϕ along a channel of type c. Note that we abstract away from the implementation of the transport layer protocol, and just model the services it provides. The channel type restricts the permissible channel endpoints: for example, if the channel was a bilateral TLS channel then the sender and receiver endpoints A_ψ and B_ϕ could not contain ?, either as a name or a channel end; conversely, if the channel was a $\textsc{tls}^{c \to s}$ channel, where the sender's name is not authenticated, then the sender's channel end would be of the form $?_\psi$. If a strand makes exclusive use of bilateral protocols then the sender's name will typically be the same on each node and would be the name contained in her certificate.

Definition 3 (Based on [8]). A *high-level term* is a tuple of the form $\sigma(X_\psi,\ Y_\phi,\ m,\ c)$ where: $\sigma \in \{+,-\}$ represents the term being sent or received respectively; $m \in \mathcal{A}$ is the application-layer message; $X_\psi \in \mathcal{I}$ is the claimed sender of m; $Y_\phi \in \mathcal{I}$ is the intended recipient of m; $c \in \mathit{Channels}$ is the channel type along which the term is communicated. Let $\hat{\mathcal{A}}$ denote the set of high-level terms. The set of finite sequences of high-level terms is denoted $\hat{\mathcal{A}}^*$.

For example, $+(?_\psi,\ B_\phi,\ m,\ \textsc{tls}^{C\to S})$ represents a message m sent along a unilateral TLS channel to an authenticated server, and $+(A_\psi,\ ?_\phi,\ m,\ \textsc{tls}^{S\to C})$ represents a message m sent along a unilateral TLS channel from the server.

Using the above we now define a strand space as follows.

Definition 4 (Based on [8]). A *strand space* is a set Σ together with a trace mapping $tr : \Sigma \to \hat{\mathcal{A}}^*$. Fix a strand space Σ.

1. A *node* is a pair $n = (st, i)$ with $st \in \Sigma$ and $i \in \{1\ ..\ |tr(st)|\}$. We say that the node n *belongs* to the strand st. The set of nodes is denoted $\mathcal{N}$. We define $msg(n) = tr(st)(i)$, and $appmsg(n) = t$ where $msg(n) = \pm(A_\psi, B_\phi, t, c)$.
2. There is an edge $n_1 \to n_2$ iff $msg(n_1) = +a$ and $msg(n_2) = -a$ for some a. This edge means that n_1 sends the term that is received by n_2.
3. There is an edge $n_1 \Rightarrow n_2$ if and only if $n_1 = (st, i)$ and $n_2 = (st, i+1)$. This edge expresses that n_1 is the immediate causal predecessor of n_2 on the strand st. The transitive closure of $\Rightarrow$ is written $\Rightarrow^+$.
4. An unsigned term t *occurs* in $n \in \mathcal{N}$ iff $t \sqsubseteq appmsg(n)$.
5. If I is a set of unsigned terms then a node $n \in \mathcal{N}$ is an *entry point* for I iff there exists $t \in I$ such that $msg(n) = +t$ and whenever $n' \Rightarrow^+ n$, $msg(n') \notin I$.
6. Unsigned term t *originates* on $n \in \mathcal{N}$ iff n is an entry point for $\{t' \mid t \sqsubseteq t'\}$.
7. The function *endpoints* $: \Sigma \to \mathcal{P}(\mathcal{I}^{reg})$ gives the set of endpoints that a regular strand uses. If n is a node on regular strand st with $msg(n) = \sigma(X_\psi,\ Y_\phi,\ m,\ c)$, then: if $\sigma = +$ then $X_\psi \in endpoints(st)$; and if $\sigma = -$ then $Y_\phi \in endpoints(st)$. Similarly, *ends* $: \Sigma \to \mathcal{P}(\mathcal{C}^{reg})$ gives the set of channel ends that a regular strand uses. Further, we assume that channel ends are partitioned by strand:
$$st \neq st' \implies ends(st) \cap ends(st') \subseteq \{?\}. \tag{1}$$

A *bundle* represents a possible real-world run of the protocol; it is a set of strands such that every message that is received by a strand is sent by another strand in the bundle.

Definition 5 (From [8]). Suppose $\to_\mathcal{B} \subset \to$, $\Rightarrow_\mathcal{B} \subset \Rightarrow$ and $\mathcal{B} = (\mathcal{N}_\mathcal{B}, \to_\mathcal{B} \cup \Rightarrow_\mathcal{B})$ is a subgraph of $(\mathcal{N}, \to \cup \Rightarrow)$. Then $\mathcal{B}$ is a *bundle* if: 1. $\mathcal{B}$ is finite; 2. if $n_2 \in \mathcal{N}_\mathcal{B}$ and $msg(n_2)$ is negative then there exists a unique $n_1 \in \mathcal{N}_\mathcal{B}$ such that $n_1 \to n_2$; 3. if $n_2 \in \mathcal{N}_\mathcal{B}$ and $n_1 \Rightarrow n_2$ then $n_1 \Rightarrow_\mathcal{B} n_2$; 4. $\mathcal{B}$ is acyclic.

Fix a bundle $\mathcal{B} = (\mathcal{N}_\mathcal{B}, \to_\mathcal{B} \cup \Rightarrow_\mathcal{B})$. A node n is *in* $\mathcal{B}$, written $n \in \mathcal{B}$, iff $n \in \mathcal{N}_\mathcal{B}$. A strand st is in $\mathcal{B}$ iff any of its nodes is in $\mathcal{N}_\mathcal{B}$. The *$\mathcal{B}$-height* of a strand st is the largest i such that $(st, i) \in \mathcal{B}$. $\prec_\mathcal{B}$ is the transitive closure of $\to_\mathcal{B} \cup \Rightarrow_\mathcal{B}$, and $\preceq_\mathcal{B}$ is the reflexive closure of $\prec_\mathcal{B}$ on $\mathcal{N}_\mathcal{B}$. $\mathcal{B}$ is *equivalent* to a bundle $\mathcal{B}'$ iff they have the same regular nodes.

The bundle below represents a client retrieving e-mail via a hypothetical protocol that makes use of unilateral TLS. It contains a client strand st_C and a server strand st_S, such that $ends(st_C) = \{\psi\}$ and $ends(st_S) = \{\phi\}$. All messages are sent over a unilateral TLS channel where the server is authenticated and the client is not; therefore, the client is identified as ? in every high-level term.

$$\begin{tikzcd}
st_C & st_S \\
\bullet \arrow[r, "(?_\psi,\, S_\phi,\, C\hat{\ }Passwd,\, TLS^{C \to S})"] \arrow[d, Rightarrow] & \bullet \arrow[d, Rightarrow] \\
\bullet & \bullet \arrow[l, "(S_\phi,\, ?_\psi,\, Messages,\, TLS^{S \to C})"']
\end{tikzcd}$$

The above example also illustrates the necessity of channel ends. Without the channel ends, a message sent along a unilateral channel to the unauthenticated end would be represented by $(S,\ ?,\ m,\ c)$. As this representation does not identify the recipient in any way it would not be possible to decide if the penetrator was allowed to receive the message. Therefore, if the channel type c provided a confidentiality guarantee, then this would admit false attacks. Hence we use channel ends to determine if the penetrator may read the contents of a message sent to an unauthenticated recipient, depending on whether the recipient endpoint is in $\mathcal{I}^{reg}$ or $\mathcal{I}^{pen}$.

The penetrator for TLS-like protocols. We now consider how to model the capabilities of the penetrator. We start with messages that are sent over the unprotected channel $\perp$ (i.e. not over any proper secure transport protocol). Clearly, we need to model the penetrator as the full Dolev-Yao penetrator and therefore we allow the penetrator to perform all the usual actions.

Definition 6 (From [8])**.** The *application-layer penetrator strands* are strands of the following form:[1]

M Text message: $\langle +(?, ?, r, \perp) \rangle$ where $r \in \mathcal{A}_{\mathcal{P}}$;
K Key: $\langle +(?, ?, k, \perp) \rangle$ where $k \in \mathcal{K}_{\mathcal{P}}$;
C Concatenation: $\langle -(?, ?, t_0, \perp), -(?, ?, t_1, \perp), +(?, ?, t_0\hat{\ }t_1, \perp) \rangle$;
S Separation: $\langle -(?, ?, t_0\hat{\ }t_1, \perp), +(?, ?, t_0, \perp), +(?, ?, t_1, \perp) \rangle$;
E Encryption: $\langle -(?, ?, k, \perp), -(?, ?, t, \perp), +(?, ?, \{t\}_k, \perp) \rangle$ where $k \in \mathcal{K}$;
D Decryption: $\langle -(?, ?, k^{-1}, \perp), -(?, ?, \{t\}_k), +(?, ?, t, \perp) \rangle$ where $k \in \mathcal{K}$.

We now consider what the penetrator can do with messages sent over a secure transport channel. For ease of exposition we firstly consider a restricted model in which the only secure transport protocols are bilateral and unilateral TLS. Clearly, we must allow the penetrator to receive messages intended for him, and to send messages coming from himself.

Definition 7 (Based on [8])**.** The *transport-layer penetrator strands* for TLS-like protocols are of the following form, where $c \neq \perp$:

SD Sending: $\langle -(?,\ ?,\ m,\ \perp), +(P_\psi,\ B_\phi,\ m,\ c) \rangle$ where $P_\psi \in \mathcal{I}^{pen}$, $B_\phi \in \mathcal{I}^{reg}$;
RV Receive: $\langle -(A_\psi,\ P_\phi,\ m,\ c), +(?,\ ?,\ m,\ \perp) \rangle$ where $P_\phi \in \mathcal{I}^{pen}$, $A_\psi \in \mathcal{I}^{reg}$.

Note that the side conditions $B_\phi \in \mathcal{I}^{reg}$ and $A_\psi \in \mathcal{I}^{reg}$ avoid redundancies caused by the penetrator sending a message to himself.

[1] We abbreviate $?_?$ to ?, for simplicity of notation.

Observe that SD strands allow the penetrator to send messages to regular strands from the client end of a unilateral TLS connection, and to claim to be some honest agent A within the application layer message, e.g. $+(?_\psi, B_\phi, A\hat{}\ldots, \textsc{tls}^{C\to S})$. However, from the server end of a unilateral TLS (or bilateral TLS) connection, he has to use a penetrator identity $P \in \mathcal{T}^{pen}_{names}$, e.g. $+(P_\psi, ?_\phi, m, \textsc{tls}^{S\to C})$, so couldn't claim to have an identity other than P within m. Similarly RV strands allow the penetrator to receive, at the client end, messages intended for regular agents, e.g. $-(A_\psi,\ ?_\phi,\ B\hat{}\ldots,\ \textsc{tls}^{S\to C})$. But at the server end, such a message would have to have a penetrator identity as the recipient to allow a RV strand, e.g. $-(?,\ P_\phi,\ P\hat{}\ldots,\ \textsc{tls}^{C\to S})$.

Further, the definition captures session properties of TLS-like protocols. The condition $P_\psi \in \mathcal{I}^{pen}$ ensures that if a regular strand receives two messages $(?_\psi,\ B_\phi,\ m,\ c)$ and $(?_\psi,\ B_\phi,\ m',\ c)$, either both came from another regular strand (if $?_\psi \in \mathcal{I}^{reg}$), or both come from penetrator SD strands (if $?_\psi \in \mathcal{I}^{pen}$).

Thus we claim that this model accurately captures the penetrator's capabilities when using TLS-like protocols. As noted in the Introduction, we have not prohibited the penetrator from reordering messages but could do so if required.

Generalising the secure transport protocol. For secure transport protocols that are weaker than TLS, there are many other ways for the penetrator to interact with transport messages. We consider how to generalise the above in order to model such protocols. In particular, we follow the approach taken in [3,8] to define a more general penetrator that can also:

- Learn a message sent to an honest endpoint (i.e. overhear);
- Fake a message as coming from an honest endpoint;
- Hijack a message, redirecting it to a different endpoint, and/or re-ascribing it as coming from a different endpoint (changing name and/or channel end).

Definition 8 (Based on [8])**.** The *transport-layer penetrator strands* are SD and RV strands, as above, and strands of the following forms, where $c \neq \perp$:

LN Learn: $\langle -(A_\psi,\ B_\phi,\ m,\ c), +(?,\ ?,\ m,\ \perp)\rangle$ where $A_\psi, B_\phi \in \mathcal{I}^{reg}$;
FK Fake: $\langle -(?,\ ?,\ m,\ \perp), +(A_\psi,\ B_\phi,\ m,\ c)\rangle$ where $A_\psi, B_\phi \in \mathcal{I}^{reg}$;
HJ Hijacking: $\langle -(X_\psi,\ Y_\phi,\ m,\ c), +(X'_{\psi'},\ Y'_{\phi'},\ m,\ c)\rangle$ providing (1) either $X_\psi \neq X'_{\psi'}$ or $Y_\phi \neq Y'_{\phi'}$; and (2) $X = ? \Leftrightarrow X' = ?$, $Y = ? \Leftrightarrow Y' = ?$, $\psi = ? \Leftrightarrow \psi' = ?$, and $\phi = ? \Leftrightarrow \phi' = ?$.

The restrictions on ? in HJ strands prevent a unilateral protocol from being transformed into a bilateral protocol, or vice-versa.

We now define channel properties that restrict the penetrator's behaviour. There are many different channel properties. We consider only the most important; other definitions could be made if specialised applications require them.

The first property we consider is *confidentiality*. Clearly LN strands should be prohibited on confidential channels. However, this is not sufficient: suppose A_ψ sends a message to B_ϕ along a confidential channel c; if the penetrator could redirect the message to $I_\chi \in \mathcal{I}^{pen}$, then he would be able to do a receive and thus can obtain the message indirectly; we therefore prohibit such behaviours.

Definition 9 (Confidential)**.** Let channel c satisfy $\mathcal{C}$. Then in any high-level bundle there are no LN strands on c, or HJ strands of the form $\langle -(X_\psi,\ Y_\phi,\ m,\ c), +(X'_{\psi'},\ Y'_{\phi'},\ m,\ c)\rangle$ where $Y_\phi \neq Y'_{\phi'}$ and $Y_\phi \in \mathcal{I}^{reg}$.

Many application-layer protocols require some guarantee that messages came from a certain source and that they were intended for a particular destination, i.e. that the channel is an *authenticated* one. Clearly, this means that fake strands must be prohibited on this channel. Furthermore, redirects and re-ascribes must also be prohibited since these allow messages to be sent to unintended destinations and for messages to have incorrect purported senders respectively.

Definition 10 (Authenticated)**.** Let channel c satisfy $\mathcal{A}$. Then no high-level bundle contains any FK or HJ strands on c.

The conjunction of the $\mathcal{A}$ and the $\mathcal{C}$ properties is known as $\mathcal{AC}$. Clearly, the only penetrator strands allowed are SD and RV strands and therefore, as discussed earlier, it corresponds to the security guarantees modelled by TLS.

3 Proof Rules

In this section we give proof rules that are of use when proving the correctness of protocols. The first, given a message that is purported to have been sent by a regular agent, allows the existence of a corresponding regular node to be deduced.

Authentication Proof Rule. Let $\mathcal{B}$ be a high-level bundle and $n \in \mathcal{B}$ be a node on a regular strand st such that $msg(n) = -(A_\psi,\ B_\phi,\ m,\ c)$ for some $A_\psi \neq ?_?$, B_ϕ, m and c. Then, providing c satisfies $\mathcal{A}$, and $A_\psi \in \mathcal{I}^{reg}$, there must exist a regular node n' such that $n' \to n$ and $msg(n') = +(A_\psi, B_\phi, m, c)$. Further, if $\phi \neq ?$ and c additionally satisfies $\mathcal{C}$[2] then every n'' such that $n' \to n''$ lies on the same strand as n.

Proof. Consider the node n' such that $n' \to n$ and suppose for a contradiction that n' is a penetrator node; then as $A_\psi \in \mathcal{I}^{reg}$ it follows that the only type of penetrator strand that n' could be on is a FK. However, these are prohibited by the assumption that c satisfies $\mathcal{A}$. Therefore n' is a regular node. Thus, as $n' \to n$, it immediately follows that $msg(n') = +(A_\psi, B_\phi, m, c)$.

Further, suppose $\phi \neq ?$, c satisfies $\mathcal{AC}$ and let n'' be a node such that $n' \to n''$. Thus, $msg(n'') = -(A_\psi, B_\phi, m, c)$ and therefore, as $B_\phi \in \mathcal{I}^{reg}$ and c satisfies $\mathcal{AC}$, n'' must be a regular node on a regular strand st''. Hence, $\phi \in ends(st'')$ and therefore Equation 1 can be applied to st'' and st to deduce that $st'' = st$. Hence n'' lies on the same strand as n, as required. □

The second proof rule extends the above rule by not only proving the existence of the regular node, but also proving that it lies on the same strand as another regular node (providing the channel ends match).

[2] In the version presented at FAST 2011 this condition was erroneously omitted.

Session Proof Rule. Let $\mathcal{B}$ be a high-level bundle and $n \in \mathcal{B}$ be a node on a regular strand *st* such that $msg(n) = -(A_\psi,\ B_\phi,\ m,\ c)$ for some $A_\psi \in \mathcal{I}^{reg}$, B_ϕ, m and c. Further, let $st' \neq st$ be another regular strand such that $A_\psi \in endpoints(st')$. Then, providing c satisfies at least $\mathcal{A}$ and $\psi \neq ?$ there must exist a regular node n' on st' such that $n' \rightarrow n$.

Proof. This follows from the previous rule by Equation 1. □

The third proof rule is mainly applicable to unilaterally authenticating secure transport channels. We first review the notion of *safe atoms* from [9].

Definition 11. The set of terms *deducible by the penetrator*, denoted $\mathcal{A}^*_\mathcal{P}$, is defined as $\{t \mid \exists t' \in \mathcal{A}_\mathcal{P} \wedge t \sqsubseteq t'\}$.

Definition 12 (From [9])**.** Let $\mathcal{B}$ be a high-level bundle. We say that a term t is *sent confidentially* in a high-level term $(A_\psi,\ B_\phi,\ m,\ c)$ iff $t \sqsubseteq m$, c satisfies $\mathcal{C}$, and $A, B \notin \mathcal{T}^{pen}_{names}$.

Definition 13 (From [9])**.** The set of *safe* atoms in a high-level bundle $\mathcal{B}$ is defined inductively by $\mathcal{M}(\mathcal{B}) = \bigcup_i \mathcal{M}_i(\mathcal{B})$ where:

- $a \in \mathcal{M}_0(\mathcal{B})$ iff $a \notin \mathcal{A}^*_\mathcal{P}$ and, if a occurs in a high-level term $m = (A_\psi,\ B_\phi,\ m,\ c)$ then a is sent confidentially in m.
- $\mathcal{M}_{i+1}(\mathcal{B}) = \mathcal{M}_i(\mathcal{B}) \cup X_{i+1}(\mathcal{B})$ where $a \in X_{i+1}(\mathcal{B})$ iff $a \notin \mathcal{A}^*_\mathcal{P}$ and if a occurs in a high-level term m then either a is sent confidentially in m or a occurs only within the set of terms $\{\{t\}_k \mid t \in \mathcal{A} \wedge k^{-1} \in \mathcal{M}_i(\mathcal{B})\}$ in m.

Lemma 14 (From [9])**.** Let $\mathcal{B}$ be a bundle and a a safe atom in $\mathcal{B}$; then for every equivalent bundle $\mathcal{B}'$ there is no node $n \in \mathcal{B}'$ such that $msg(n) = (?,\ ?,\ a,\ \perp)$.

Authentication via Confidentiality Proof Rule. Let $\mathcal{B}$ be a bundle and a be a safe atom in $\mathcal{B}$. Furthermore, let $n \in \mathcal{B}$ be a regular node such that $msg(n) = -(A_\psi, B_\phi, m, c)$ for some A_ψ, B_ϕ, m and c, such that $m = \ldots\hat{\ }a\hat{\ }\ldots$, and c satisfies $\mathcal{AC}$. Then there exists a regular node $n' \in \mathcal{B}$ such that $n' \rightarrow n$.

Proof. Let n' be the node such that $n' \rightarrow n$. Suppose for a contradiction that n' is a penetrator node. Then as c satisfies $\mathcal{AC}$ it follows that n' must lie on a SD strand and thus there must exist nodes n_1 and n_2 such that $n_2 \rightarrow n_1 \Rightarrow n'$ and $msg(n_2) = +(?,\ ?,\ m,\ \perp)$. Hence there exists an equivalent bundle $\mathcal{B}'$ that contains the same regular nodes, together with extra S strands to extract a from m to obtain a node n'' such that $msg(n'') = (?,\ ?,\ a,\ \perp)$. However, a is safe so this contradicts Lemma 14. Therefore n' must be a regular node. □

4 The WebAuth Protocol

In this section we introduce the WebAuth protocol and describe some of the unusual issues that arise when verifying web-based protocols. Then we formally define a strand space for the WebAuth protocol, and state our assumptions.

WebAuth [11] is a *single-sign on protocol* that is designed to allow users to login to multiple websites through a central authentication server, meaning that only one username and password per user is required. It differs from other single-sign on protocols such as OpenID [12] in that it requires shared keys to be established between the website and the authentication server prior to authentication of users.

Three principals participate in a WebAuth session: the *User Agent* (UA), is the web browser that makes requests for the user; the *Application Server* (AS) is the server that the user wishes to access; the *Login Server* (LS) is the server responsible for authenticating the user. These principals communicate via HTTP [13] or HTTPS [14] requests, and pass data to each other by embedding *tokens* in the redirect URLs. For example, when the AS redirects the UA to the LS, the redirect URL will be of the form `https://`*`LS`*`/?RT=`*`rtok`*`;ST=`*`stok`* where *rtok* and *stok* are tokens. The AS and LS also use HTTP cookies to store tokens to allow the user agent to re-authenticate on subsequent requests.

In this paper we prove the correctness of the *initial sign on* mode of WebAuth, which assumes that the user is not already authenticated. The protocol in its most simplified form is as follows:

$1.\ UA \rightarrow AS : Request$
$2.\ AS \rightarrow UA : RequestToken\hat{}\,ServiceToken\hat{}\,LS$
$3.\ UA \rightarrow LS\ : RequestToken\hat{}\,ServiceToken$
$4.\ LS \rightarrow UA : LoginForm\hat{}\,RequestToken\hat{}\,ServiceToken$
$5.\ UA \rightarrow LS\ : RequestToken\hat{}\,ServiceToken\hat{}\,U\hat{}\,passwd_{LS}(U)$
$6.\ LS \rightarrow UA : AS\hat{}\,Request\hat{}\,ProxyToken\hat{}\,IdToken$
$7.\ UA \rightarrow AS : Request\hat{}\,IdToken$
$8.\ AS \rightarrow UA : Response\hat{}\,AppToken$

A *RequestToken* encapsulates the original request that the user made. The *ServiceToken* contains configuration information for the LS, enabling it to be stateless. The *ProxyToken* allows a user to authenticate again without supplying her password (i.e. *repeat authentication*), whilst the *IdToken* is a temporary token created by the LS for the AS that details who the user is. The user exchanges this temporary token for a *AppToken* by passing it to the AS.

WebAuth's token encoding is complicated, so we use a simplified version; essentially the same proof would hold for the full protocol. We encode a token by $\{\mathsf{tag}\hat{}\,data\}_{key}$ where $key \in \mathcal{K}_{sym}$ and tag is a tag. The protocol can be described as follows, where $SK(A)$ denotes a symmetric key that is secret to $A \in \mathcal{T}_{names}$ and Sh^{AS}_{LS} denotes the key shared between AS and LS.

$1.\ UA \rightarrow AS : r$
$2.\ AS \rightarrow UA : LS\hat{}\{\mathsf{req}\hat{}\,r\}_{Sh^{AS}_{LS}}\hat{}\{\mathsf{webkdc\_service}\hat{}\,AS\hat{}\,Sh^{AS}_{LS}\}_{SK(LS)}$
$3.\ UA \rightarrow LS\ : \{\mathsf{req}\hat{}\,r\}_{Sh^{AS}_{LS}}\hat{}\{\mathsf{webkdc\_service}\hat{}\,AS\hat{}\,Sh^{AS}_{LS}\}_{SK(LS)}$
$4.\ LS \rightarrow UA : LoginForm\hat{}\{\mathsf{req}\hat{}\,r\}_{Sh^{AS}_{LS}}\hat{}\{\mathsf{webkdc\_service}\hat{}\,AS\hat{}\,Sh^{AS}_{LS}\}_{SK(LS)}$
$5.\ UA \rightarrow LS\ : U\hat{}\,passwd_{LS}(U)\hat{}\{\mathsf{req}\hat{}\,r\}_{Sh^{AS}_{LS}}\hat{}\{\mathsf{webkdc\_service}\hat{}\,AS\hat{}\,Sh^{AS}_{LS}\}_{SK(LS)}$
$6.\ LS \rightarrow UA : AS\hat{}\,r\hat{}\{\mathsf{webkdc\_proxy}\hat{}\,U\}_{SK(LS)}\hat{}\{\mathsf{id}\hat{}\,U\}_{Sh^{AS}_{LS}}$
$7.\ UA \rightarrow AS : r\hat{}\{\mathsf{id}\hat{}\,U\}_{Sh^{AS}_{LS}}$
$8.\ AS \rightarrow UA : resp\hat{}\{\mathsf{app}\hat{}\,U\hat{}\,Sh^{AS}_{LS}\}_{SK(AS)}.$

WebAuth mandates the use of HTTPS between *LS* and either *AS* or *UA*, but merely *recommends* that HTTPS is used between *UA* and *AS*. Clearly, if HTTPS is not used between *UA* and *AS* then there are a number of attacks whereby the intruder intercepts various tokens. Thus, we assume that all messages are sent over unilateral TLS, with *UA* unauthenticated.

When modelling security protocols it is generally assumed that the participants are able to perform checks on the values they receive to ensure adherence to the protocol. For example, a *UA* may be expected to compare the value of r received in message 6 to the one sent in message 1. However, these checks are not possible if the role is being assumed by a general-purpose web browser. In particular this means that the user will not check if the request and the *AS* match between messages 1 and 7, or if the request and service tokens match.

Further, the servers are *stateless*. For example, the AS stores no state between the first two messages and the last two; we will therefore model these two exchanges using two distinct strands (and similarly for the LS).

We now define the strand space corresponding to the protocol. In the following: ψ_i denotes channel ends used by the user; ϕ_i denotes channel ends used by the *LS* and *AS*; r_i denotes requests; *stok*, *rtok*, *atok*, *ptok*, *idtok* denote service, request, application, proxy and identity tokens respectively. Further, we assume that the set of atoms, $\mathcal{T}$, includes requests, responses, the token tags, passwords and the login form (denoted by *LoginForm*).

Definition 15. A *Web-Auth Strand Space* consists of the union of the images of the following functions.

$AS1[AS, LS, \psi_1, \phi_1, r_1, stok] \mathrel{\widehat{=}}$ // Messages 1, 2

$\quad \langle -(?_{\psi_1},\ AS_{\phi_1},\ r_1,\ \textsc{tls}^{C\to S}), +(AS_{\phi_1},\ ?_{\psi_1},\ \{\mathsf{req}\,\hat{}\,r_1\}_{Sh^{AS}_{LS}}\,\hat{}\,stok,\ \textsc{tls}^{S\to C})\rangle$

$AS2[AS, U, LS, \psi_4, \phi_4, r_2, resp] \mathrel{\widehat{=}}$ // Messages 7, 8

$\quad \langle -(?_{\psi_4},\ AS_{\phi_4},\ r_2\,\hat{}\,\{\mathsf{id}\,\hat{}\,U\}_{Sh^{AS}_{LS}},\ \textsc{tls}^{C\to S}),$

$\quad + (AS_{\phi_4},\ ?_{\psi_4},\ resp\,\hat{}\,\{\mathsf{app}\,\hat{}\,U\,\hat{}\,Sh^{AS}_{LS}\}_{SK(AS)},\ \textsc{tls}^{S\to C})\rangle$

$LS1[LS, AS, \psi_2, \phi_2, k, r_2] \mathrel{\widehat{=}}$ // Messages 3, 4

$\quad \langle -(?_{\psi_2},\ LS_{\phi_2},\ \{\mathsf{req}\,\hat{}\,r_2\}_k\,\hat{}\,\{\mathsf{webkdc\_service}\,\hat{}\,AS\,\hat{}\,k\}_{SK(LS)},\ \textsc{tls}^{C\to S}),$

$\quad + (LS_{\phi_2},\ ?_{\psi_2},$

$\qquad LoginForm\,\hat{}\,\{\mathsf{req}\,\hat{}\,r_2\}_k\,\hat{}\,\{\mathsf{webkdc\_service}\,\hat{}\,AS\,\hat{}\,k\}_{SK(LS)},\ \textsc{tls}^{S\to C})\rangle$

$LS2[LS, U, AS, \psi_3, \phi_3, k, r_2] \mathrel{\widehat{=}}$ // Messages 5, 6

$\quad \langle -(?_{\psi_3}, LS_{\phi_3},$

$\qquad U\,\hat{}\,passwd_{LS}(U)\,\hat{}\,\{\mathsf{req}\,\hat{}\,r_1\}_k\,\hat{}\,\{\mathsf{webkdc\_service}\,\hat{}\,AS\,\hat{}\,k\}_{SK(LS)}, \textsc{tls}^{C\to S}),$

$\quad + (LS_{\phi_3},\ ?_{\psi_3},\ AS\,\hat{}\,r\,\hat{}\,\{\mathsf{proxy}\,\hat{}\,U\}_{SK(LS)}\,\hat{}\,\{\mathsf{id}\,\hat{}\,U\}_{Sh^{AS}_{LS}},\ \textsc{tls}^{S\to C})\rangle$

$User[U, AS, AS', LS, \psi_1, \psi_2, \psi_3, \psi_4, \phi_1, \phi_2, \phi_3, \phi_4, r_1, r_2, resp, rtok_1, rtok_2,$

$\qquad stok_1, stok_2, pt, idtok, atok] \mathrel{\widehat{=}}$

$\quad \langle +(?_{\psi_1},\ AS_{\phi_1},\ r_1,\ \textsc{tls}^{C\to S}), -(AS_{\phi_1},\ ?_{\psi_1},\ LS\,\hat{}\,rtok_1\,\hat{}\,stok_1,\ \textsc{tls}^{S\to C}),$

$+ (?_{\psi_2},\ LS_{\phi_2},\ rtok_1\hat{}\,stok_1,\ \textsc{tls}^{C\to S}),$
$- (LS_{\phi_2},\ ?_{\psi_2},\ LoginForm\hat{}\,rtok_2\hat{}\,stok_2,\ \textsc{tls}^{S\to C}),$
$+ (?_{\psi_3},\ LS_{\phi_3},\ U\hat{}\,passwd_{LS}(U)\hat{}\,rtok_2\hat{}\,stok_2,\ \textsc{tls}^{C\to S}),$
$- (LS_{\phi_3},\ ?_{\psi_3},\ AS'\hat{}\,r_2\hat{}\,pt\hat{}\,idtok,\ \textsc{tls}^{S\to C}),$
$+ (?_{\psi_4},\ AS'_{\phi_4},\ r_2\hat{}\,idtok,\ \textsc{tls}^{C\to S}), -(AS'_{\phi_4},\ ?_{\psi_4},\ resp\hat{}\,atok,\ \textsc{tls}^{S\to C})\rangle$

In order to prove the correctness of WebAuth we require a number of assumptions: 1. the penetrator does not initially know any key shared between two honest agents; 2. honest application servers are configured with the correct service tokens and keys; 3. the only non-atomic terms known to the penetrator are service tokens for dishonest servers; 4. the penetrator does not initially know passwords of honest users.

Assumption 16. 1. If $A \in \mathcal{T}^{reg}_{names}$ then $SK(A) \notin \mathcal{A}_{\mathcal{P}}$, and if $A, B \in \mathcal{T}^{reg}_{names}$ then $Sh^A_B \notin \mathcal{A}_{\mathcal{P}}$; 2. If $st \in AS1[AS, LS, \psi_1, \phi_1, r_1, stok]$ then $stok = \{\mathsf{webkdc\_service}\hat{}\,AS\hat{}\,Sh^{AS}_{LS}\}_{SK(LS)}$; 3. The only non-atomic terms in $\mathcal{A}_{\mathcal{P}}$ are of the form $\{\mathsf{webkdc\_service}\hat{}\,AS\hat{}\,Sh^{AS}_{LS}\}_{SK(LS)}$ for $AS \in \mathcal{T}^{pen}_{names}$; 4. If $U, LS \in \mathcal{T}^{reg}_{names}$ then $passwd_{LS}(U) \notin \mathcal{A}_{\mathcal{P}}$.

Further, we require that the user does not reveal her password except to the appropriate login server; i.e. the user is not tricked into giving her password away to the penetrator. In practice this means that the user, before divulging her password, should verify the LS by ensuring that the domain name matches the expected name; this requirement may not be obvious to all users. This assumption is formalised in the definition of the strand space: in message 6 on a *User* strand, $(?_{\psi_3},\ LS_{\phi_3},\ U\hat{}\,passwd_{LS}(U)\hat{}\,rtok_2\hat{}\,stok_2,\ \textsc{tls}^{C\to S})$, the identities of the recipient and of the server in $passwd_{LS}(U)$ are required to be equal.

5 The Guarantees

We start by proving that shared keys and passwords are safe.

Lemma 17. Let $\mathcal{B}$ be a bundle from Σ. If $AS \in \mathcal{T}^{reg}_{names}$ and $LS \in \mathcal{T}^{reg}_{names}$ then Sh^{AS}_{LS} is a safe key. Further, if $U \in \mathcal{T}^{reg}_{names}$ and $LS \in \mathcal{T}^{reg}_{names}$ then $passwd_{LS}(U)$ is safe.

Proof. The latter follows from Assumption 16 and the fact that $passwd_{LS}(U)$ is always sent confidentially in $\mathcal{B}$ (as $\textsc{tls}^{C\to S}$ satisfies $\mathcal{C}$). For the former, by Assumption 16, $SK(LS) \notin \mathcal{A}_{\mathcal{P}}$; further, by the definition of Σ, $SK(LS)$ does not appear as a subterm of any message. Therefore, $SK(LS)$ is a safe key. Assuming $AS \in \mathcal{T}^{reg}_{names}$ it follows by Assumption 16 that $Sh^{AS}_{LS} \notin \mathcal{A}_{\mathcal{P}}$. Further, since Sh^{AS}_{LS} appears only as a subterm of messages encrypted using a safe key (i.e. $SK(LS)$) it follows that Sh^{AS}_{LS} is a safe key. □

We now analyse what the user can deduce having completed a full run of the protocol. The proposition and its proof are illustrated in Figure 1.

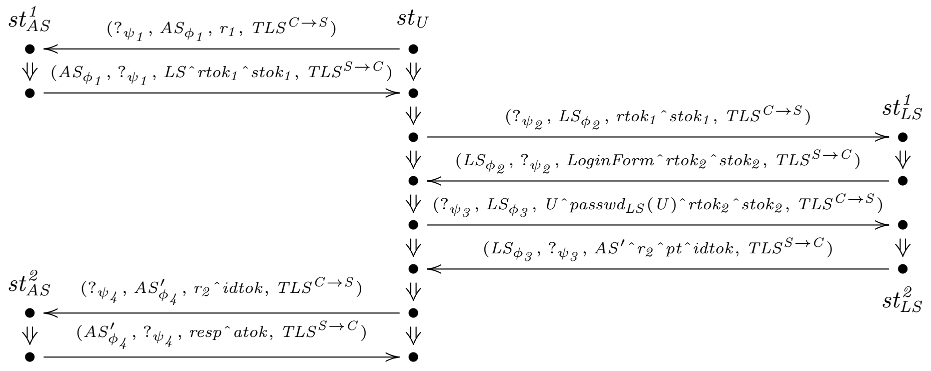
st_{AS}^{1}
st_U
st_{LS}^{1}
$(?_{\psi_1}, AS_{\phi_1}, r_1, TLS^{C\to S})$
$(AS_{\phi_1}, ?_{\psi_1}, LS\hat{}rtok_1\hat{}stok_1, TLS^{S\to C})$
$(?_{\psi_2}, LS_{\phi_2}, rtok_1\hat{}stok_1, TLS^{C\to S})$
$(LS_{\phi_2}, ?_{\psi_2}, LoginForm\hat{}rtok_2\hat{}stok_2, TLS^{S\to C})$
$(?_{\psi_3}, LS_{\phi_3}, U\hat{}passwd_{LS}(U)\hat{}rtok_2\hat{}stok_2, TLS^{C\to S})$
$(LS_{\phi_3}, ?_{\psi_3}, AS'\hat{}r_2\hat{}pt\hat{}idtok, TLS^{S\to C})$
st_{AS}^{2}
$(?_{\psi_4}, AS'_{\phi_4}, r_2\hat{}idtok, TLS^{C\to S})$
st_{LS}^{2}
$(AS'_{\phi_4}, ?_{\psi_4}, resp\hat{}atok, TLS^{S\to C})$

Fig. 1. A graphical illustration of Proposition 18.

Proposition 18. Let $\mathcal{B}$ be a bundle from Σ and let

$$st_U \in User[U, AS, AS', LS, \psi_1, \psi_2, \psi_3, \psi_4, \phi_1, \phi_2, \phi_3, \phi_4, r_1, r_2, resp, rtok_1, rtok_2, stok_1, stok_2, pt, idtok, atok]$$

be a regular strand of $\mathcal{B}$-height 8. Then provided $LS \in \mathcal{T}_{names}^{reg}$:

1. If $AS \in \mathcal{T}_{names}^{reg}$ then there exists a strand $st_{AS}^{1} \in AS1[AS, LS, \psi_1, \phi_1, r_1, stok_1]$ of $\mathcal{B}$-height 2 such that $st_U(1) \to st_{AS}^{1}(1)$, $st_{AS}^{1}(2) \to st_U(2)$ and $rtok_1 = \{\mathsf{req}\hat{}r_1\}_{Sh_{LS}^{AS}}$;
2. There exists a strand $st_{LS}^{1} \in LS1[LS, AS'', \psi_2, \phi_2, k, r_2']$ of $\mathcal{B}$-height 2 such that $st_U(3) \to st_{LS}^{1}(1)$ and $st_{LS}^{1}(2) \to st_U(4)$; further $stok_1 = stok_2 = \{\mathsf{webkdc\_service}\hat{}Sh_{LS}^{AS''}\}_{SK(LS)}$ and $rtok_1 = rtok_2 = \{\mathsf{req}\hat{}r_2'\}_{Sh_{LS}^{AS''}}$;
3. There exists a strand $st_{LS}^{2} \in LS2[LS, U, AS'', \psi_3, \phi_3, k, r_2']$ of $\mathcal{B}$-height 2 such that $st_U(5) \to st_{LS}^{2}(1)$, $st_{LS}^{2}(2) \to st_U(6)$, $r_2 = r_2'$, $AS' = AS''$, $pt = \{\mathsf{proxy}\hat{}U\}_{SK(LS)}$ and $idtok = \{\mathsf{id}\hat{}U\}_{Sh_{LS}^{AS''}}$;
4. If $AS' \in \mathcal{T}_{names}^{reg}$ then there exists a strand $st_{AS}^{2} \in AS2[AS', U, LS, \psi_4, \phi_4, r_2, resp]$ of $\mathcal{B}$-height 2 such that $st_U(7) \to st_{AS}^{2}(1)$, $st_{AS}^{2}(2) \to st_U(8)$ and $atok = \{\mathsf{app}\hat{}U\hat{}Sh_{LS}^{AS'}\}_{SK(LS)}$;
5. If $AS \in \mathcal{T}_{names}^{reg}$ then $r_1 = r_2$ and $AS = AS'$;
6. If $resp \notin \mathcal{A}_{\mathcal{P}}$ and $AS' \in \mathcal{T}_{names}^{reg}$ then $resp$ occurs safely in $\mathcal{B}$;
7. The strands st_{AS}^{1}, st_{AS}^{2}, st_{LS}^{1} and st_{LS}^{2} so defined are unique.

Proof. Let st_U be such a strand; we prove each of the points in turn as follows:

1. Assume that $AS \in \mathcal{T}_{names}^{reg}$. As $\textsc{tls}^{S\to C}$ satisfies $\mathcal{A}$, by the Authentication Rule there must exist a regular node n such that $n \to st_U(2)$, and thus that $msg(n) = msg(st_U(2))$. By inspection this can only be the second node on an $AS1$ strand st_{AS}^{1}. Consider the node n' such that $n' \to st_{AS}^{1}(1)$; since $\textsc{tls}^{C\to S}$ satisfies $\mathcal{A}$, the Session Rule can be applied to $st_{AS}^{1}(1)$ to deduce that n' must be regular and lie on st_U. By inspection this can only be the first node. Hence $msg(st_U(1)) = msg(st_{AS}^{1}(1))$ and therefore, $st_{AS}^{1} \in AS1[AS, LS, \psi_1, \phi_1, r_1, stok_1]$, and hence $rtok_1 = \{\mathsf{req}\hat{}r_1\}_{Sh_{LS}^{AS}}$.

2. Again, as $\textsc{tls}^{s\to c}$ satisfies $\mathcal{AC}$ and since $LS_{\phi_2} \in \mathcal{I}^{reg}$ it follows, by the Authentication Rule, that there must exist a regular node n such that $n \to st_U(4)$, and thus that $msg(n) = msg(st_U(4))$. By inspection this can only be the second node on a $LS1$ strand st^1_{AS}. Also, by the Session Rule there exists a regular node n' on st_U such that $n' \to st^1_{LS}(1)$. By inspection this can only be $st_U(3)$, and therefore $st^1_{LS} \in LS1[LS, AS'', \psi_2, \phi_2, k, r'_2]$ for some AS'' and k. Therefore, it immediately follows that $rtok_2 = rtok_1 = \{\mathsf{req}\,\hat{}\,r'_2\}_{Sh^{AS'}_{LS}}$, and $stok_2 = stok_1 = \{\mathsf{webkdc\_service}\,\hat{}\,Sh^{AS'}_{LS}\}_{SK(LS)}$.
3. The proof of this case is similar to before and shows that $st^2_{LS} \in LS2[LS, U, AS'', \psi_3, \phi_3, k, r'_2]$, $AS'' = AS'$ and $r'_2 = r_2$.
4. The proof of this case is similar to before and shows that $st^2_{AS} \in AS2[AS', U, LS, \psi_4, \phi_4, r_2, resp]$.
5. This follows from part 1 and part 3.
6. Note that $\textsc{tls}^{s\to c}$ satisfies $\mathcal{C}$ and therefore, provided $AS' \in \mathcal{T}^{reg}_{names}$, $resp$ is sent confidentially in $st^2_{AS}(2)$. Therefore, it immediately follows that $resp$ occurs safely providing $resp \notin \mathcal{A}_{\mathcal{P}}$.
7. This follows immediately from Equation 1 as disjoint regular strands use disjoint channel ends. □

We now consider the guarantees to the login server. We require a lemma that shows that only correct keys can be embedded in service tokens. The proof is in Appendix A.

Lemma 19. Let $\mathcal{B}$ be a bundle from Σ, $LS \in \mathcal{T}^{reg}_{names}$ and $st^2_{LS} \in LS2[LS, U, AS, \psi_3, \phi_3, k, r_2]$ be a regular strand of $\mathcal{B}$-height at least 1. Then $k = Sh^{AS}_{LS}$.

Proposition 20. Let $\mathcal{B}$ be a bundle from Σ, $LS \in \mathcal{T}^{reg}_{names}$ and $st^2_{LS} \in LS2[LS, U, AS', \psi_3, \phi_3, k, r_2]$ be a regular strand of $\mathcal{B}$-height 2. Then:

1. If $AS' \in \mathcal{T}^{reg}_{names}$ then there exists a strand $st^1_{AS} \in AS1[AS', LS, \psi_1, \phi_1, r_2, stok]$ of $\mathcal{B}$-height 2;
2. If $U \in \mathcal{T}^{reg}_{names}$ then there exists a strand (writing $*$ for values that are arbitrary) $st_U \in User[U, AS, *, LS, \psi'_1, \psi_2, \psi_3, *, \phi'_1, \phi_2, \phi_3, *, r_1, *, *, rt, rt, stok_1, stok_1, *, *, *]$ of $\mathcal{B}$-height at least 5, and there exists a strand $st^1_{LS} \in LS1[LS, AS', \psi_2, \phi_2, k, r_2]$, such that $st_U(3) \to st^1_{LS}(1)$, $st^1_{LS}(2) \to st_U(4)$, $st_U(5) \to st^2_{LS}(1)$ and $st^2_{LS}(2) \to st_U(6)$;
3. If $AS, U \in \mathcal{T}^{reg}_{names}$ then $st_U(1) \to st^1_{AS}(1)$, $st^1_{AS}(2) \to st_U(2)$, $\psi_1 = \psi'_1$, $\phi_1 = \phi'_1$, $stok = stok_1$, $AS = AS'$ and $r_1 = r_2$.

The proof uses the following proof rule from [9], based on [15].

Incoming Authentication Test. Suppose that there is a negative node $n_1 \in \mathcal{B}$ such that $msg(n_1) = (A_\psi,\ B_\phi,\ m,\ c)$, $t = \{t_0\}_K \sqsubseteq m$ and $K \in \mathcal{M}(\mathcal{B})$. If $t \notin \mathcal{A}^*_{\mathcal{P}}$ then there exists a regular node $n'_1 \prec_{\mathcal{B}} n_1$ such that t originates on n'_1.

Proof sketch of Proposition 20. 1. Using the fact that $k = Sh^{AS'}_{LS}$ is safe, the Incoming Authentication Test applied to $st^2_{LS}(1)$, can be used to show the node that created the request token must be regular; hence an appropriate $AS1$ strand exists. 2. By the Authentication via Confidentiality Rule, using the fact that

$\mathit{passwd}_{LS}(U)$ is safe. 3. All participants are honest, so U and LS agree on where the request token originates from, which must therefore be the $\mathit{AS1}$ strand from part 1. The full proof is in Appendix A. □

Lastly, we consider what an application server (in particular, an $\mathit{AS2}$) can deduce having completed a run of the protocol.

Proposition 21. Let $\mathcal{B}$ be a bundle from Σ, $AS', LS \in \mathcal{T}_{names}^{reg}$ and $st_{AS}^{2} \in \mathit{AS2}[AS', U, LS, \psi_4, \phi_4, r_2, \mathit{resp}]$ be a regular strand of $\mathcal{B}$-height 2. Then:

1. There exists a strand $st_{LS}^{2} \in \mathit{LS2}[LS, U, AS', \psi_2, \phi_2, k, r_2']$ of $\mathcal{B}$-height 2;
2. If $U \in \mathcal{T}_{names}^{reg}$ then:
   (a) There exists a strand $st_U \in \mathit{User}[U, AS, AS', LS, \psi_1, \psi_2, \psi_3, \psi_4, \phi_1, \phi_2, \phi_3, \phi_4, r_1, r_2, *, rt, rt, \mathit{stok}, \mathit{stok}, pt, \mathit{idtok}, \mathit{atok}]$ of $\mathcal{B}$-height at least 7;
   (b) $r_2' = r_2$;
   (c) If $AS \in \mathcal{T}_{names}^{reg}$ then $AS = AS'$ and there exists a strand $st_{AS}^{1} \in \mathit{AS1}[AS, LS, \psi_1, \phi_1, r_1, \mathit{stok}]$ of $\mathcal{B}$-height 2;
   (d) There exists a strand $st_{LS}^{1} \in \mathit{LS1}[LS, AS', \psi_2, \phi_2, k, r_2]$ of $\mathcal{B}$-height 2;
   (e) $st_U(3) \to st_{LS}^{1}(1)$, $st_{LS}^{1}(2) \to st_U(4)$, $st_U(5) \to st_{LS}^{2}(1)$, $st_{LS}^{2}(2) \to st_U(6)$, $st_U(7) \to st_{AS}^{2}(1)$; and if $AS \in \mathcal{T}_{names}^{reg}$ then $st_U(1) \to st_{AS}^{1}(1)$, $st_{AS}^{1}(2) \to st_U(2)$, and $r_1 = r_2$.

Proof sketch. 1. From the Incoming Authentication Test applied to $st_{AS}^{2}(1)$. 2. From an extension of Proposition 20. The full proof is in Appendix A. □

Item 2c reveals a subtlety of the protocol: the application server has no guarantee that the user wishes to authenticate herself to it. For example, suppose there are two application servers, one dishonest, P, and one honest, AS. Further, suppose the user wishes to access a resource on P; when P redirects the user to the Login Server, rather than P sending his own service and request token, P can send a service token for AS and a request token for a resource r on AS. This means that the user, after successfully authenticating to LS, would be redirected to AS and would inadvertently request r. Clearly, this could be dangerous if r is a request that causes data to be modified.

6 Conclusions

In this paper we have modified the high-level strand spaces model to model the security guarantees provided by unilaterally authenticating secure transport protocols. Further, the use of channel ends allows us to capture fine-grained session properties. This alteration makes proofs easier to develop and comprehend compared to the model of [8,9].

We have also provided general proof rules which, when used in conjunction with the Authentication Tests of [9], give a general proof strategy for proving the correctness of application layer protocols that use either unilateral or bilateral secure transport protocols. In particular, given a strand st, the proof rules from Section 3 can be applied to show the existence of strands that *directly* send to st.

This is in contrast to the Authentication Tests, which can be used to show the existence of strands that send *indirectly* to *st* (i.e. via another strand).

We demonstrated the model by proving the correctness of a single-sign on protocol, WebAuth. Whilst this analysis did not reveal any major attacks it did reveal a requirement on the user to check that she has been redirected to the correct Login Server, and one subtlety, that the application server has no guarantee that the user wishes to authenticate herself to it.

Another problem that arises when verifying protocols where one of the participants is a web browser is that the web browser does not check if messages are skipped or reordered. For example, there is nothing to prevent a dishonest application server from sending a message 4 rather than a message 2. In principle, it would be possible to adapt the proofs to model this behaviour, but the proofs would be rather intricate and uninteresting.

Related work. The work that is closest to ours is that of Mödersheim and Viganò [7]. They define a model, the *Ideal Channel Model (ICM)*, that abstracts away from how the channels are implemented. They then consider how to model the guarantees given by unilaterally authenticating secure transport protocols (or *secure pseudonymous channels*). In their model they specify confidential, authentic and secure channels, which roughly correspond to $\mathcal{C}$, $\mathcal{A}$ and $\mathcal{AC}$ respectively. The primary difference between the two approaches is that whilst both formalisms permit analysis of protocols that use bilateral or unilateral transport protocols, ours also allows protocols that do not group messages into sessions to be analysed (i.e. by letting the channel end be ?). Another difference is that they address unauthenticated clients using *pseudonyms* rather than our name and channel end combination, which we use to enable bilateral and unilateral protocols to be considered together more uniformly. We also think that this clarifies the model and makes it clearer what is occurring at the transport layer. Further, they do not consider session properties.

One interesting difference is in how the proofs of the application layer protocols are developed. The formalism of Mödersheim and Viganò requires the prover to explicitly detail what constitutes an attack. This contrasts with our formalism where the exact correctness properties are deduced *during* the proof; we believe that this offers an advantage as there may be many small, subtle, correctness conditions.

In [16] Groß and Mödersheim consider *vertical protocol composition*, which occurs, for example, when a TLS connection is layered on a secure VPN connection. In particular, they develop a composition result that proves that, providing each protocol satisfies certain conditions, arbitrary composition of the set of protocols introduces no new attacks.

Future work. In this paper we considered how to prove the correctness of application-layer protocols that are layered upon secure transport protocols. Clearly we need to justify the correctness of the model in order to ensure the proofs are valid. In particular we need to prove that unilateral TLS satisfies $\mathcal{AC}$, which could be done by adapting the proof for bilateral TLS from [17]. Further, in order to show that the model is sound we need to relate the abstract model to the concrete model where the transport protocol and application-layer protocol are combined together; this could be done by adapting the proof from [18].

In order to make our technique more applicable it would be interesting to consider methods by which it could be automated. One approach to this problem would be to adapt the tool CPSA [19], which is able to analyse protocols in the standard strand spaces model. Alternatively, since proofs using our model are largely mechanical we believe that a proof assistant would be easy to develop.

It would also be interesting to consider what further enhancements could be made to the penetrator model to enable additional secure transport properties to be developed. For example, we may wish to model secure transport protocols that ensure messages are received in the correct order.

Acknowledgements We would like to thank the anonymous reviewers for useful comments.

References

1. T. Dierks and E. Rescorla, “The TLS Protocol: Version 1.2.” `http://tools.ietf.org/html/rfc5246`, 2008.
2. G. Bella, C. Longo, and L. Paulson, “Verifying Second-Level Security Protocols,” in *Theorem Proving in Higher Order Logics*, 2003.
3. C. Dilloway and G. Lowe, “Specifying Secure Transport Channels,” in *Computer Security Foundations Symposium*, 2008.
4. A. Armando, R. Carbone, and L. Compagna, “LTL Model Checking for Security Protocols,” in *Computer Security Foundations Symposium*, 2007.
5. 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 *Formal Methods in Security Engineering*, 2008.
6. M. Bugliesi and R. Focardi, “Language Based Secure Communication,” in *Computer Security Foundations Symposium*, 2008.
7. S. Mödersheim and L. Viganò, “Secure pseudonymous channels,” in *European Symposium on Research in Computer Security*, 2009.
8. A. Kamil and G. Lowe, “Specifying and Modelling Secure Channels in Strand Spaces,” in *Formal Aspects in Security and Trust*, 2009.
9. A. Kamil, *The Modelling and Analysis of Layered Security Architectures in Strand Spaces.* DPhil Thesis, University of Oxford, 2010.
10. F. J. Thayer Fábrega, J. Herzog, and J. Guttman, “Strand spaces: Proving Security Protocols Correct,” *Journal of Computer Security*, 1999.
11. R. Schemers and R. Allbery, “WebAuth Technical Specification v.3.” `http://webauth.stanford.edu/protocol`, 2009.
12. B. Fitzpatrick, D. Recordon, D. Hardt, J. Bufu, and J. Hoyt, “OpenID Authentication 2.0.” `http://openid.net/specs/openid-authentication-2_0.html`.
13. P. J. Leach, T. Berners-Lee, J. C. Mogul, L. Masinter, R. T. Fielding, and J. Gettys, “Hypertext Transfer Protocol – HTTP/1.1.” `http://tools.ietf.org/html/rfc2616`, 2004.
14. E. Rescorla, “HTTP Over TLS.” `http://tools.ietf.org/html/rfc2818`, 2000.

15. J. D. Guttman and F. J. Thayer, "Authentication tests and the structure of bundles," *Theoretical Computer Science*, 2002.
16. T. Groß and S. Mödersheim, "Vertical Protocol Composition," in *Computer Security Foundations Symposium*, 2011.
17. A. Kamil and G. Lowe, "Analysing TLS in the Strand Spaces Model," *To appear in Journal of Computer Security*, 2011.
18. A. Kamil and G. Lowe, "Understanding Abstractions of Secure Channels," in *Formal Aspects of Security and Trust*, 2010.
19. S. Doghmi, J. Guttman, and F. J. Thayer, "Searching for Shapes in Cryptographic Protocols," in *Tools and Algorithms for the Construction and Analysis of Systems*, 2007.

A Extra Proofs from Section 5

Proof of Lemma 19. Firstly, note that by Assumption 16, $SK(LS) \notin \mathcal{A}_{\mathcal{P}}$; further it never appears as a subterm of a message. Hence, it follows trivially that $SK(LS)$ can never appear as the key edge on a E strand meaning that the penetrator cannot use $SK(LS)$ as an encryption key. Therefore, the penetrator can never construct a term of the form $\{\mathsf{webkdc\_service}\hat{\ }AS\hat{\ }k\}_{SK(LS)}$. Thus, it follows that terms of this form must either be in $\mathcal{A}_{\mathcal{P}}$, and hence of the correct form by Assumption 16, or originate from regular AS stands and hence, due to Assumption 16, be of the correct form. □

Proof of Proposition 20. Let st_{LS}^{2} be such a strand.

1. By Lemma 19, $k = Sh_{LS}^{AS'}$; and by Lemma 17, $Sh_{LS}^{AS'}$ is safe. Therefore, the Incoming Authentication Test can be applied to $st_{LS}^{2}(1)$ to deduce that there must exist a regular node $n \prec st_{LS}^{2}(1)$ such that $\{\mathsf{req}\hat{\ }r_2\}_{Sh_{LS}^{AS'}}$ originates on n. By inspection this can only be the second node on an $AS1$ strand $st_{AS}^{1} \in AS1[AS', LS, \psi_1, \phi_1, r_2, stok]$.
2. Assuming that $U \in \mathcal{T}_{names}^{reg}$ it follows by Lemma 17 that $passwd_{LS}(U)$ is safe. Therefore, by the Authentication via Confidentiality Rule it follows that there exists a regular node n' such that $n' \to st_{LS}^{2}(1)$. By inspection, this can only be the fifth node on a user strand, st_U:

$$msg(st_U(5)) = +(?_{\psi_3},\ LS_{\phi_3},\ U\hat{\ }passwd_{LS}(U)\hat{\ }rtok_2\hat{\ }stok_2,\ \textsc{tls}^{C\to S}).$$

Therefore $st_U \in User[U, AS, *, LS, \psi_1', \psi_2, \psi_3, *, \phi_1', \phi_2, \phi_3, *, r_1, *, *, rtok_1, rtok_2, stok_1, stok_2, *, *, *]$. Further, as $LS \in \mathcal{T}_{names}^{reg}$, by the Authentication Rule there exists a regular node n such that $msg(n) = msg(st_U(4))$. By inspection, this can only be the second node on a $LS1$ strand, st_{LS}^{1}. Hence, it follows that $\psi_2 \in ends(st_U(4))$ and thus that $?_{\psi_2} \in \mathcal{I}^{reg}$. Therefore, by the Session Rule applied to $st_{LS}^{1}(1)$ there exists a regular node n' on st_U such that $msg(n') = msg(st_{LS}^{1}(1))$. Furthermore, the only node of the correct form is $st_U(3)$, and hence it follows that $st_{LS}^{1} \in LS1[LS, AS', \psi_2, \phi_2, k, r_2]$, and that $stok_1 = stok_2$ and $rtok_1 = rtok_2$.

3. As $AS, U \in \mathcal{T}^{reg}_{names}$, it is possible to prove, in identical fashion to Point (1) of Proposition 18, that there exists a strand ${st^{1}_{AS}}' \in \mathit{AS1}[AS, LS, \psi'_{1}, \phi'_{1}, r_{1}, \mathit{stok_1}]$. Therefore, by Assumption 16 it immediately follows that $\mathit{stok_1} = \{\mathsf{webkdc\_service} \hat{} AS \hat{} k\}_{SK(LS)}$ and thus that $AS' = AS$. Furthermore, taking this and the results of Point (2) above, we have shown (writing m_i for each high-level term) that the following situation occurs:

$$\begin{tikzcd}
st_U & {st^{1}_{AS}}' & st^{1}_{LS} & st^{2}_{LS} \\
\bullet \arrow[r, "m_1"] & \bullet \arrow[d, Rightarrow] & & \\
\bullet \arrow[d, Rightarrow] & \bullet \arrow[l, "m_2"'] & & \\
\bullet \arrow[rr, "m_3"] & & \bullet \arrow[d, Rightarrow] & \\
\bullet \arrow[d, Rightarrow] & & \bullet \arrow[ll, "m_4"'] & \\
\bullet \arrow[rrr, "m_5"] & & & \bullet \arrow[d, Rightarrow] \\
{} & & & \bullet \arrow[lll, "m_6"']
\end{tikzcd}$$

Recall in the proof of Point (1) above we showed that $st^{1}_{AS}(2) \prec st^{2}_{LS}(1)$. Therefore, it follows that $st^{1}_{AS}(2)$ must be one of the above nodes. Thus, as it lies on an $\mathit{AS1}$ strand it must be the case that $st^{1}_{AS} = {st^{1}_{AS}}'$. Therefore $\psi_1 = \psi'_1$, $\phi_1 = \phi'_1$, $\mathit{stok} = \mathit{stok_1}$ and $r_1 = r_2$ as required. □

Proof of Proposition 21. Let st^{2}_{LS} be such a strand.

1. As $AS', LS \in \mathcal{T}^{reg}_{names}$, it follows by Lemma 17 that $Sh^{AS'}_{LS}$ is a safe key. Hence, by the Incoming Authentication Test applied to $st^{2}_{AS}(1)$ there must exist a regular node $n \prec st^{2}_{AS}(1)$ such that $\{\mathsf{id} \hat{} U\}_{Sh^{AS'}_{LS}} \sqsubseteq \mathit{msg}(n)$. By inspection, this can only be the second node on a $\mathit{LS2}$ strand, and hence there exists a strand $st^{2}_{LS} \in \mathit{LS2}[LS, U, AS', \psi_3, \phi_3, k, r'_2]$.
2. This follows immediately from a trivial extension of Proposition 20 to a *User* strand of $\mathcal{B}$-height 7 (we need to extend the *User* strand to ensure agreement between the *User* and $\mathit{LS2}$ on r_2 and r'_2). □