Chaining Secure Channels

Christopher Dilloway
Oxford University Computing Laboratory

Abstract

Security architectures often make use of secure transport protocols to protect network messages: the transport protocols provide secure channels between hosts. In this paper we examine the possibilities for chaining secure channels. We present a surprising theorem that shows that, in some cases, two channels can be chained through a proxy to produce a stronger channel. We also show that the channel established through a proxy is at least as strong as the greatest lower bound of the channels established to and from the proxy.

Keywords Formal specification, security models, secure channels, proxies, chaining.

1 Background

A popular technique for designing a security architecture is to rely on a secure transport layer to protect messages on the network, and provide secure channels between different hosts; see e.g. [8, 10, 11]. In [4] we describe a framework for specifying the sorts of security guarantees that might be provided by secure channels. We capture security properties using CSP-style trace specifications, building on the work of Broadfoot and Lowe [2].

In [4] we exclusively describe channels that secure point-to-point connections. In this paper we examine two different ways in which secure channels can be chained: first through a set of dedicated intermediaries (simple proxies), and then through a set of trustworthy (multiplexing) proxies. We present a surprising theorem that shows how, under some circumstances, two channels can be chained to produce a stronger channel. We also show that the channel established through a proxy is always at least as strong as the greatest lower bound of the channels established to and from the proxy.

The results presented in this paper are particularly relevant to real-world use of secure channels. Many organisations arrange their computers in a trusted intranet, and only allow external access through a proxy. In order to perform its role (e.g. virus scanning, or traffic management), a proxy must be able to establish secure channels on behalf of the internal agents.

It is not obvious which security properties, if any, are provided by establishing secure channels through a proxy. The aim of this paper, therefore, is to investigate these sorts of chains of channels in order to determine which security properties the overall channel can provide.

In Section 2 we describe the model of secure channel specifications from [4]. In Section 3 we present the first chaining theorem; in this section the proxies

are dedicated intermediaries. Every agent knows, and can verify, the identity of their own proxies, and also of the proxies who send messages to them on other agents' behalf. This public knowledge of the job of each simple proxy restricts the intruder's behaviour, and so, in some cases, two channels can be chained to produce a stronger channel.

In Section 4 we describe the second chaining theorem; in this section the proxies are more general. Each proxy accepts messages from any agent, and is willing to forward them to any other agent. The agents and the proxies add extra information to the application-layer messages that they send in order to identify the third party involved in the message (the original sender, or the intended recipient). This extra information limits the intruder's behaviour, and so, as before, two channels can be chained to produce a stronger channel. The proofs of the two chaining theorems are omitted from this paper (for lack of space), but can be found in [3].

Finally, in Section 5 we discuss related work, and in Section 6 we discuss the utility of the results presented in this paper, and we present some ideas for future work.

2 Secure channels

In [4] we define an abstract network in terms of honest agents, who send and receive messages, and an intruder, who has several events he can use to manipulate the messages being passed on the network, and who can also send and receive messages.

Our model is split into two layers: in the application layer the agents play roles, establish channels, and send and receive messages; in the secure transport layer protocol agents translate the higher level events into lower level events (e.g. by encrypting or signing messages), and vice versa (e.g. by decrypting messages or verifying signatures). The agents and protocol agents refer to the channels they establish by local connection identifiers.

A channel is described by an ordered pair of roles; for example, the channel (R_i, R_j) (which we often write as $R_i \rightarrow R_j$) is the channel in which agents playing role R_i send messages to agents playing role R_j.

We treat encryption formally. All messages are drawn from the message space, *Message*, which is partitioned into two sets: application-layer messages ($Message_{App}$) and transport-layer messages ($Message_{TL}$). We assume a monotonic and transitive relation $\vdash$ defined over this type: for $X \subseteq Message$, and $m : Message$, $X \vdash m$ means that m can be deduced from the set X. We assume that the intruder has some initial knowledge $IIK \subseteq Message$.

We use the following events, where m ranges over the set $Message_{App}$ of application-layer messages.

send$.(A, R_i).c_A.(B, R_j).m$**:** the agent (A, R_i)[1] sends message m, intended for agent (B, R_j), in a connection identified by A as c_A.

receive$.(B, R_j).c_B.(A, R_i).m$**:** the agent (B, R_j) receives message m, apparently from agent (A, R_i), in a connection identified by B as c_B.

fake$.(A, R_i).(B, R_j).c_B.m$**:** the intruder fakes a *send* of message m to agent (B, R_j) in connection c_B with the identity of honest agent (A, R_i).

[1] The notation (A, R_i) refers to the agent with identity A playing role R_i

hijack$.(A, R_i) \rightarrow (A', R_i).(B, R_j) \rightarrow (B', R_j).c_{B'}.m$**:** the intruder modifies a previously sent message m and changes the sender to (A', R_i), and the receiver to (B', R_j) so that B' accepts it in connection $c_{B'}$.

By changing both, just one, or neither of the identities associated with a message, the intruder can use the *hijack* event in four different ways: to *replay* a previously-sent message, to *re-ascribe*[2] a message, to *redirect* a message, or to *re-ascribe and redirect* a message. In some cases the intruder can only perform these events with a dishonest identity.

In [4] we specify several rules that define the application-layer behaviour accepted by our networks, and define the relationship between the application-layer events and the transport-layer events performed by the honest agents. These rules do not capture channel properties; rather, they define some sanity conditions in order to remove artificial and irrelevant behaviour from our networks.

A channel specification is a predicate over the set of valid system traces: the (prefix-closed) set of traces that satisfy the network rules referred to above; this set is dependent upon the intruder's initial knowledge. A channel specification has a natural interpretation: the set of valid system traces that it accepts, assuming some value of the intruder's initial knowledge.

A confidential channel should protect the confidentiality of any message sent on it from all but the intended recipient. For example, a confidential channel can be implemented by encrypting application layer messages with the intended recipient's public key. We identify confidential channels by tagging them with the label C (e.g. writing $C(R_i \rightarrow R_j)$). The intruder's knowledge is then restricted so that he only learns from messages that are sent on non-confidential channels, or that are sent to him. We note that this notion of confidentiality is that of Dolev and Yao [5]: the intruder can only decrypt messages when he possesses the decryption key; we do not attempt to capture any other definition of confidentiality (such as indistinguishability).

We specify authenticated channels by describing the relationship between the *receive* and *send* events performed by the agents at either end of the channel. In particular, we specify under what circumstances an agent may perform a particular *receive* event. The bottom of our hierarchy is the standard Dolev-Yao network model, captured by the network rules in [4].

There are two dishonest events the intruder can perform: faking and hijacking. With some transport protocols the latter can only be performed using dishonest identities. We specify our channels by placing restrictions on when he can perform these events.

No faking: $\boldsymbol{NF}(R_i \rightarrow R_j)$**:** the intruder cannot fake messages on the channel.

No-re-ascribing: $\boldsymbol{NRA}(R_i \rightarrow R_j)$**:** the intruder cannot change the sender's identity when he hijacks messages on the channel.

No-honest-re-ascribing: $\boldsymbol{NRA}^{-}(R_i \rightarrow R_j)$**:** the intruder can only change the sender's identity to a dishonest identity when he hijacks messages on the channel.

No-redirecting: $\boldsymbol{NR}(R_i \rightarrow R_j)$**:** the intruder cannot redirect messages on the channel.

[2] To ascribe means to attribute a text to a particular person; hence we use "re-ascribe" to describe the intruder's activity when he changes the identity of the sender of a message.

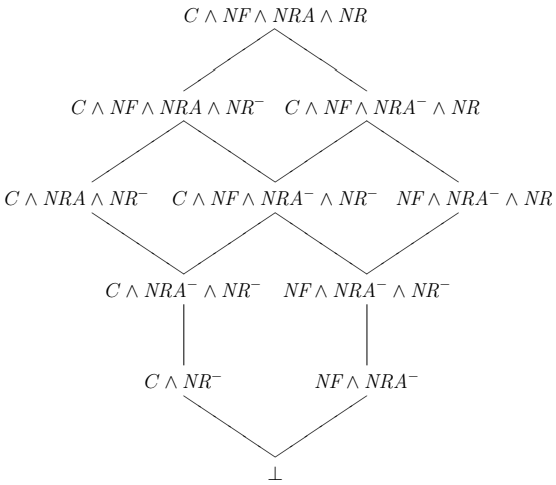

Figure 1: The hierarchy of secure channels.

No-honest-redirecting: $NR^-(R_i \to R_j)$: the intruder cannot redirect messages that were sent to honest agents on the channel.

The formal specifications simply block the relevant events; for example:

$$NR^-(R_i \to R_j)(tr) \mathrel{\widehat{=}} tr \downarrow \{\mid hijack.A \to A'.B \to B' \mid A, A' : \hat{R}_i; B, B' : \hat{R}_j \cdot B \neq B' \wedge Honest(B) \mid\} = \langle\rangle \,.$$

The formal specifications for all of the properties are shown in Appendix A.1.

These properties are not independent, since no-re-ascribing implies no-honest-re-ascribing, and likewise for no-redirecting. Further, not all combinations are fundamentally different; certain pairs of combinations allow essentially the same intruder behaviours: each simulates the other. We collapse such combinations; for full details of the collapsing cases see [4]. After taking the collapsing cases into consideration we arrive at a hierarchy of four non-confidential and seven confidential channels, shown in Figure 1.

Without taking the collapsing cases into account, there are thirty-six different combinations of the primitives described above; these form a lattice. We describe a point in this lattice by listing each of its components in the order (C, NF, NRA, NR); e.g. the point $(C, \bot, NRA, NR^-)$ is the channel $C \wedge NRA \wedge NR^-$. There is a partial order on the lattice: points are compared component-wise.

The collapsing cases, which are described in detail in [4], can also be described by five collapsing rules:

$$\begin{aligned}
\mathcal{C}_1 &\mathrel{\widehat{=}} (\bot,\ \bot,\ x,\ y) \downarrow (\bot,\ \bot,\ \bot,\ \bot)\,,\\
\mathcal{C}_2 &\mathrel{\widehat{=}} (x,\ NF,\ \bot,\ y) \downarrow (x,\ \bot,\ \bot,\ y)\,,\\
\mathcal{C}_3 &\mathrel{\widehat{=}} (\bot,\ NF,\ NRA,\ x) \downarrow (\bot,\ NF,\ NRA^-,\ x)\,,\\
\mathcal{C}_4 &\mathrel{\widehat{=}} (C,\ x,\ y,\ \bot) \downarrow (\bot,\ x,\ y,\ \bot)\,,\\
\mathcal{C}_5 &\mathrel{\widehat{=}} (C,\ \bot,\ x,\ NR) \downarrow (C,\ \bot,\ x,\ NR^-)\,.
\end{aligned}$$

Definition 2.1. For any point (c, nf, nra, nr) in the full lattice, we define $\downarrow (c, nf, nra, nr)$ to be the *collapsed form* of the point. This is the point we

reach by continually applying the collapsing rules until we reach a point that cannot be collapsed further.

Proposition 2.2. *The collapsed form of every point in the full lattice is unique and well-defined.*

The proof of this proposition is a simple case analysis; for details see [3]. We note that $\downarrow$ is monotonic with respect to the order on the lattice.

In [4] we define a simulation relation that compares channel specifications by comparing the honest agents' views of them. The honest agents' view of the traces of a channel specification is the restriction of those traces to the application-layer send and receive events performed by the honest agents.

The channel specification $\mathit{ChannelSpec}_1$ simulates $\mathit{ChannelSpec}_2$ if, for all possible values of the intruder's initial knowledge, every trace of the second specification corresponds to a trace of the first specification that appears the same to the honest agents. We write $\mathit{ChannelSpec}_1 \preccurlyeq \mathit{ChannelSpec}_2$.

3 Simple proxies

In this section we present the chaining theorem for *simple proxies*.

Definition 3.1. A *simple proxy* is an agent who is dedicated to forwarding messages from one agent to another. For every pair of roles (R_i, R_j) there is a simple proxy role $\mathit{Proxy}_{(R_i,R_j)}$. For every pair of agents $(A : \hat{R}_i, B : \hat{R}_j)$[3] such that $A \neq B$ there is a unique simple proxy $P_{(A,B)} : \widehat{\mathit{Proxy}}_{(R_i,R_j)}$ who forwards messages from A to B. When two roles communicate through a simple proxy, the following properties hold:

- Agents playing roles R_i and R_j do not communicate directly;
- For every pair of agents (A, B), the simple proxy $P_{(A,B)}$ only establishes connections with A and B, and each connection it establishes is either dedicated to sending messages to B or to receiving messages from A. Further, the simple proxy forwards to B every message that it receives from A;
- Honest agents only send messages to their own proxies, and they only receive messages from proxies whom they know send messages to them.

Because the proxy $P_{(A,B)}$ acts on A's behalf, the proxy is honest if and only if A is honest. We think of the family of proxies $\{P_{(A,B)} \mid B : \mathit{Agent}\}$ as A's proxies (because they all send on her behalf).

3.1 Secure channels through simple proxies

In order to discover which security properties the channel through a simple proxy satisfies we consider each of the components of the hierarchy individually. In the discussion below we refer to the channel to the proxy as $(R_i \rightarrow \mathit{Proxy}_{(R_i,R_j)})$, and the channel from the proxy as $(\mathit{Proxy}_{(R_i,R_j)} \rightarrow R_j)$.

[3]The notation $\hat{R}_i$ is shorthand for $(\mathit{Identity}, R_i)$.

Confidentiality It is clear that if either of the channels to or from a simple proxy is not confidential, the channel through the proxy is not confidential.

No faking In order to fake a message from A to B, the intruder must either fake sending the message to A's proxy $P_{(A,B)}$, or fake sending the message from A's proxy to B; the channel through the proxy is fakeable if either the channel to the proxy or the channel from the proxy is fakeable.

No re-ascribing The intruder can either re-ascribe a message on the channel to the proxy or on the channel from the proxy.

In order to re-ascribe a message on the channel to the proxy it is not enough for the intruder just to change the sender's identity ($\mathit{hijack}.A \rightarrow A'.P_{(A,B)}.c_P.m$) because A's proxy will not accept a message that appears to have been sent by another agent (A'). In order to re-ascribe a message on the channel to the proxy, the intruder must also be able to redirect the message to the correct one of the new sender's proxies.

On the other hand, re-ascribing a message on the channel from the proxy is straightforward:[4] $\mathit{hijack}.P_{(A,B)} \rightarrow P_{(A',B)}.B.c_B.m$. The intruder only needs to change the sender's identity to that of another agent's proxy.

No redirecting The intruder can either redirect a message on the channel to the proxy or on the channel from the proxy.

In order to redirect a message on the channel to the proxy the intruder simply redirects to a different proxy: $\mathit{hijack}.A.P_{(A,B)} \rightarrow P_{(A,B')}.c_P.m$.

On the other hand, in order to redirect a message on the channel from the proxy the intruder cannot just change the recipient's identity ($\mathit{hijack}.P_{(A,B)}.B \rightarrow B'.c_{B'}.m$). B' will not accept a message from the proxy $P_{(A,B)}$ because B' knows which proxies send messages to him; in order to redirect a message on this channel the intruder must also be able to re-ascribe the message to one of the proxies that B' accepts messages from.

If roles R_i and R_j are communicating through simple proxies, then Definition 3.1 prevents agents playing role R_i from communicating directly with agents playing role R_j: it insists that they only communicate through proxies. This means that the standard definitions of our secure channels (which restrict the intruder's behaviour when hijacking or faking messages) are vacuously satisfied: there are no messages sent by agents playing role R_i to agents playing role R_j to hijack, and no agent playing role R_j will accept a message that appears to be from an agent playing role R_i.

We have seen that in order to fake a message, the intruder can fake it on the channel to the proxy, or on the channel from the proxy. We have also seen that the intruder can hijack messages on either the channel to the proxy, or on the channel from the proxy. In order to block these activities, we must do so on both channels; for example:

$$\begin{aligned} &\mathit{NF}(\mathit{Proxy}_{(R_i,R_j)})(tr) \mathrel{\widehat{=}} \\ &\quad tr \downarrow \{\!|\, \mathit{fake}.\hat{R}_i.\widehat{\mathit{Proxy}}_{(R_i,R_j)}, \mathit{fake}.\widehat{\mathit{Proxy}}_{(R_i,R_j)}.\hat{R}_j \,|\!\} = \langle\rangle\,. \end{aligned}$$

The definitions of all of our building blocks on the channel through a simple proxy are shown in A.2.

[4] Note that the proxies $P_{(A,B)}$ and $P_{(A',B)}$ are different agents.

3.2 Simple chaining theorem

We make two observations of the overall channel through a simple proxy.

Observation 3.2. If the intruder cannot redirect messages that were sent to honest agents on the channel to the proxy, then he cannot re-ascribe messages on the channel to the proxy. In order to re-ascribe a message the intruder must be able to redirect the message to one of the new sender's proxies. Further, since all honest agents' proxies are honest, no honest agent ever sends a message to a dishonest agent on the channel to the proxy. Subject to the collapsing cases described earlier, if the channel to the proxy satisfies NR^- it also satisfies $NRA \wedge NR$.

Observation 3.3. If the intruder cannot re-ascribe messages to honest agents on the channel from the proxy, then he cannot redirect messages on the channel from the proxy. In order to redirect a message the intruder must be able to re-ascribe it to one of the proxies who sends messages to the new recipient. Subject to the collapsing cases described earlier, if the channel from the proxy satisfies NRA^- it also satisfies $NRA^- \wedge NR$.

Theorem 3.4 (Simple chaining theorem)**.** *If roles R_i and R_j communicate through simple proxies on secure channels such that $ChannelSpec_1(R_i \to Proxy_{(R_i,R_j)})$, and $ChannelSpec_2(Proxy_{(R_i,R_j)} \to R_j)$, where $ChannelSpec_1$ and $ChannelSpec_2$ are channels in the hierarchy, then the overall channel (through the proxy) satisfies the channel specification $ChannelSpec = \downarrow (\nwarrow_s ChannelSpec_1 \sqcap \nearrow_s ChannelSpec_2)$, where:*

$$\nwarrow_s (c, nf, nra, nr) = \begin{array}{l} (c, nf, 2, 2) \text{ if } nr \geqslant 1\,, \\ (c, nf, nra, nr) \text{ otherwise}\,; \end{array} \qquad \nearrow_s (c, nf, nra, nr) = \begin{array}{l} (c, nf, nra, 2) \text{ if } nra \geqslant 1\,, \\ (c, nf, nra, nr) \text{ otherwise}\,; \end{array}$$

and $\sqcap$ is the greatest lower bound operator in the full lattice.

Corollary 3.5. *If roles R_i and R_j communicate through simple proxies on secure channels such that: $ChannelSpec(R_i \to Proxy_{(R_i,R_j)})$, and $ChannelSpec(Proxy_{(R_i,R_j)} \to R_j)$, where ChannelSpec is a channel in the hierarchy, then the overall channel (through the proxy) satisfies a channel specification $ChannelSpec'$ such that $ChannelSpec \preccurlyeq ChannelSpec'$. In particular, $ChannelSpec(Proxy_{(R_i,R_j)})$ holds.*

Example 3.6. The channel to the proxy satisfies $C \wedge NRA \wedge NR^-$, and the channel from the proxy satisfies $NF \wedge NRA^- \wedge NR$.

$$\nwarrow_s (C \wedge NRA \wedge NR^-) = C \wedge NRA \wedge NR\,,$$
$$\nearrow_s (NF \wedge NRA^- \wedge NR) = NF \wedge NRA^- \wedge NR\,.$$

The greatest lower bound is $NRA^- \wedge NR$, which collapses to $\bot$. The channel to the proxy is fakeable and the channel from the proxy is non-confidential; the overall channel simulates the bottom channel.

Example 3.7. The channel to the proxy satisfies $NF \wedge NRA^- \wedge NR^-$, and the channel from the proxy satisfies $NF \wedge NRA^-$.

$$\nwarrow_s (NF \wedge NRA^- \wedge NR^-) = NF \wedge NRA \wedge NR\,,$$
$$\nearrow_s (NF \wedge NRA^-) = NF \wedge NRA^- \wedge NR\,.$$

The greatest lower bound is $NF \wedge NRA^{-} \wedge NR$, which does not collapse. This channel is stronger than both of the individual channels. The intruder cannot fake messages on this channel, nor can he redirect messages (because he can't redirect messages using the channel to the proxy, and he can't re-ascribe messages using the channel from the proxy). The intruder can only re-ascribe messages with his own identity because this is the greatest capability he has on each channel individually.

3.3 An automated proof of the simple chaining theorem

Each instance of Theorem 3.4 is relatively simple to prove. One simply starts with a receive event, and calculates which events are allowed (by the channel specifications to and from the proxy, and by the network rules) to precede it in a valid system trace. Each receive event can be traced back to a set of send, fake, or hijack events; it is then straightforward to determine the strongest channel whose alternative specification is satisfied.

However, while each instance of the theorem can be proved simply, there are 121 instances[5] that must be proved. In order to ease this process we have developed a Haskell script that performs the proofs automatically; the full Haskell script is listed in [3]. In the rest of this section we give a brief description of the script; this automated proof technique is also used to prove the multiplexing chaining theorem (described in the next section).

Deriving the full set of trace patterns We first calculate the distinct trace patterns that result in an honest agent receiving a message, via a proxy, from another honest agent or the intruder. A trace pattern is a subtrace consisting of the events leading up to a receive event in which all identities, connection identifiers and message values are representative. For example, a trace pattern may show that an honest agent sends a message to their proxy, the proxy receives it and then sends it on, the intruder then redirects the message to another honest agent, and then the new recipient receives the message; e.g.

$$s \mathrel{\widehat{=}} \langle \mathit{send}.A.c_A.P_{(A,B)},\ \mathit{receive}.P_{(A,B)}.c_P.A.m,\ \mathit{send}.P_{(A,B)}.c'_P.B.m,\\ \mathit{hijack}.P_{(A,B)} \rightarrow P_{(A,B')}.B \rightarrow B'.c_{B'}.m,\ \mathit{receive}.B'.c_{B'}.P_{(A,B')}.m \rangle\ .$$

Applying the channel properties We apply the properties of the channels to and from the proxy to eliminate those trace patterns in which the intruder must perform an event that the channel does not allow him to. For example, if the intruder cannot fake on the channel to the proxy, we eliminate those trace patterns in which he fakes a message on this channel.

Determining the resultant channel specification We examine the remaining trace patterns to determine which capabilities the intruder still has. For example, if one of the remaining trace patterns shows that an honest agent A sent a message to an honest agent B, but then B receives that message from the dishonest agent I, then this pattern demonstrates that the intruder can re-ascribe messages with his own identity. When we examine each of the trace patterns we discover which events the intruder can perform; we then find the point in the lattice that corresponds to these remaining events, and collapse this point to a channel in the hierarchy.

[5] There are 11 possibilities for the channel to the proxy, and 11 for the channel from the proxy.

By eliminating trace patterns and calculating the resultant point in this manner we prove that the specification of the resultant channel holds on all valid system traces in which the channel specifications to and from the proxy hold. The list of resultant channels for every instance of the simple chaining theorem is shown in Appendix B.1. For an example proof of an instance of the theorem, and more details on the automated proof, see [3].

4 Multiplexing proxies

In this section we consider a more general proxy case. The study of simple proxies shows that by chaining two secure channels through a trusted third party one can sometimes produce a stronger channel. However, in the simple case, we thought of the proxies as 'belonging' to one of the agents communicating. In this section we consider more general multiplexing proxies. A multiplexing proxy is a trusted third party who is willing to forward messages from any agent to any other agent.

We assume that all multiplexing proxies are honest. There is nothing to stop the intruder from setting up proxies of his own; however, any message sent through a dishonest proxy cannot remain confidential, and any message received from a dishonest proxy cannot be authenticated.

When agent A intends to send a message to another agent (B) through a simple proxy she just needs to pick the correct simple proxy to send the message to. The proxy knows whom to forward the message to because it is dedicated to that job. If A is to use a multiplexing proxy, she must communicate her intent (to talk to B) to the proxy. Similarly, when B receives a message from A's proxy, he knows who originally sent the message; when B receives a message from a multiplexing proxy, there must be some communication from the proxy to B to say whom the message is from.

One way to solve this problem would be to build a special transport-layer protocol in which the message sender's protocol agent tells the proxy whom to establish a connection with, and the proxy tells the recipient's protocol agent whom the messages are from. However, this solution is unsuitable for our model: the whole point of the model is to make the details of the transport-layer protocol abstract. Once we start to impose conditions on the transport-layer protocol, we lose the generality of the abstract model.

The solution we adopt, therefore, is to annotate the application-layer messages with information about whom they are intended for, and whom they were originally sent by. In order to send a message m to B (via the multiplexing proxy P), agent A concatenates B's identity to m: $\mathit{send}.A.c_A.P.\langle m, B\rangle$. When P receives this message he concatenates it to A's identity, and sends it on to B: $\mathit{send}.P.c_P.B.\langle A, m\rangle$. This only works if the channel is either confidential or non-fakeable; however, all of our channels satisfy at least one of these two properties, so this method can be used on all of our channels.

We assume that none of the application-layer protocols call for agents to send messages with the same type as the messages described above. If we do not make this assumption, it might be possible for messages created by honest agents for use in the application-layer protocols to be mistaken for messages sent to or from a multiplexing proxy.

Definition 4.1. A *multiplexing proxy* is an honest agent who is dedicated to

forwarding messages; there is a single proxy role *Proxy*. When two roles communicate through a multiplexing proxy, the following properties hold:

- Agents playing roles R_i and R_j do not communicate directly;
- Honest agents only send messages of the form $\langle m, B \rangle$ to proxies, and only receive messages of the form $\langle A, m \rangle$ from proxies;
- Each connection that the multiplexing proxies establish is either dedicated to receiving messages from one agent, or sending messages to one agent. Further, the multiplexing proxies reliably forward every message that they receive.

Although two individual messages from one agent to another could be sent through different proxies, we assume that all the messages in one connection are sent to (or received from) the same proxy. Each multiplexing proxy can be used by several agents.

4.1 Secure channels through multiplexing proxies

The public knowledge of the role of each simple proxy was what lead to the rather surprising result that the chained form of two channels can be stronger than both channels individually. With the multiplexing proxies we no longer have this public knowledge; B only knows whom the message was originally sent by by examining it and seeing whose identity is attached to it. As we did last time, we consider each of the components of the hierarchy individually in order to discover which properties the channel through a proxy satisfies. In the discussion below we refer to the channel to the proxy as $(R_i \rightarrow \mathit{Proxy})$ and the channel from the proxy as $(\mathit{Proxy} \rightarrow R_j)$.

Confidentiality Since all multiplexing proxies are honest, the channel through the proxy is confidential if and only if the channels to and from the proxy are confidential.

No faking In order to fake a message from A to B, the intruder must either fake sending the message to the proxy or from the proxy, so if either channel is fakeable, the channel through the proxy is fakeable.

No re-ascribing Unlike the simple proxies, the intruder cannot choose which channel to re-ascribe a message on: he must do so on the channel to the proxy; this is straightforward: $\mathit{hijack}.A \rightarrow A'.P.c_P.\langle m, B \rangle$.

The only identity that the intruder can change by re-ascribing on the channel from the proxy is that of the message sender (the proxy): $\mathit{hijack}.P \rightarrow P'.B.c_B.\langle A, m \rangle$. Because honest agents only accept messages of the form $\langle A, m \rangle$ from proxies, the intruder can only re-ascribe the message to a different proxy: he cannot change the identity of the original sender of the message by re-ascribing the message on the channel from the proxy.

No redirecting The intruder can only redirect a message using the channel from the proxy; this is straightforward: $\mathit{hijack}.P.B \rightarrow B'.c_{B'}.\langle A, m \rangle$. The only identity that the intruder can change by redirecting the message on the channel to the proxy is that of the message recipient (the proxy): $\mathit{hijack}.A.P \rightarrow P'.c_{P'}.\langle m, B \rangle$. Because the only honest agents who receive messages of the form $\langle m, B \rangle$ are proxies, the intruder can only redirect the message to a different proxy.

The *Proxies* property on the roles R_i and R_j prevents agents playing role R_i from communicating directly with agents playing role R_j. As before, we must reframe the definitions of the authenticated channel building blocks for the channel through a multiplexing proxy because the standard definitions are vacuously satisfied; the definitions are shown in Appendix A.3.

4.2 Chaining theorem

We make two observations of the channel through a multiplexing proxy.

Observation 4.2. The intruder cannot redirect messages using the channel to the proxy. Subject to the collapsing cases described earlier, the channel to the proxy satisfies *NR*.

Observation 4.3. The intruder cannot re-ascribe messages using the channel from the proxy. Subject to the collapsing cases described earlier, the channel from the proxy satisfies *NRA*.

Theorem 4.4 (Chaining theorem)**.** *If roles R_i and R_j communicate through multiplexing proxies on secure channels such that $ChannelSpec_1(R_i \to Proxy)$, and $ChannelSpec_2(Proxy \to R_j)$, where $ChannelSpec_1$ and $ChannelSpec_2$ are channels in the hierarchy, then the overall channel (through the proxy) satisfies the channel specification $ChannelSpec = \downarrow (\nwarrow_m ChannelSpec_1 \sqcap \nearrow_m ChannelSpec_2)$, where:*

$$\nwarrow_m (c, nf, nra, nr) = (c, nf, nra, 2)\,, \quad \nearrow_m (c, nf, nra, nr) = (c, nf, 2, nr)\,,$$

and $\sqcap$ is the greatest lower bound operator in the full lattice.

Corollary 4.5. *If roles R_i and R_j communicate through multiplexing proxies on secure channels such that $ChannelSpec(R_i \to Proxy)$, and $ChannelSpec(Proxy \to R_j)$, where ChannelSpec is a channel in the hierarchy, then the overall channel (through the proxy) satisfies ChannelSpec.*

Example 4.6. The channel to the proxy satisfies $C \wedge NF \wedge NRA^- \wedge NR^-$, and the channel from the proxy satisfies $C \wedge NRA^- \wedge NR^-$.

$$\begin{aligned} \nwarrow_m (C \wedge NF \wedge NRA^- \wedge NR^-) &= C \wedge NF \wedge NRA^- \wedge NR\,, \\ \nearrow_m (C \wedge NRA \wedge NR^-) &= C \wedge NRA \wedge NR^-\,. \end{aligned}$$

The greatest lower bound is $C \wedge NRA^- \wedge NR^-$; this is the greatest lower bound of the two channels. This is the same result as the simple proxies. The intruder cannot re-ascribe messages to honest agents because the channel from the proxy is only re-ascribable with dishonest identities; even though the channel from the proxy is fakeable, both channels are confidential, so the intruder cannot learn the message and fake it to effect a re-ascribe. The intruder can redirect messages that are sent to him.

The list of resultant channels for every instance of the chaining theorem is shown in Appendix B.2. In order to prove this theorem we adapt the automated proof of the simple chaining theorem; for details, and for a proof of one instance of the theorem, see [3].

5 Related work

The discussion of proxies presented in this paper is based on the work of Dilloway and Lowe in [4]. Other authors have specified secure channels in different ways, and to different ends, and in some cases have demonstrated similar chaining results.

In [7], the authors describe a calculus for secure channel establishment. They define channels that offer confidentiality ($\rightarrow\bullet$), authentication of the message sender ($\bullet\rightarrow$), or both ($\bullet\rightarrow\bullet$). The authors show that if user B trusts a third party T, and there are channels from another agent A such that $A \bullet\!\!\rightarrow T \bullet\!\!\rightarrow B$, then the agents A and B can establish a new channel $A \bullet\!\!\rightarrow B$. The authors also show that confidential channels can be chained, provided that the message sender trusts the third party. These two results agree with our chaining theorems; though our results go further as we show that many more channels can be chained. However, we cannot reason about channels when only one agent trusts the third party, as the authors of [7] can.

In [1], Boyd defines two different types of channel: *Confidentiality*, where only the intended user (or set of users) can read the message; and *Authentication*, where only the expected user (or set of users) can write the message. In Boyd's setup channels are established by utilising existing channels, or by propagating new channels between the two users wishing to communicate, often via a trusted third party (a proxy in our notation). Boyd shows that if a user A has an authenticated channel to a third party T, and T has an authenticated channel to a user B (and if B trusts T), then an authenticated channel from A to B can be established. This agrees with our (multiplexing) result that authenticated channels can be chained; as before though, our results go further as they show that many more channels can be chained.

Some authors have tried to solve the chaining problem by modifying the secure transport layer protocol. In [9] the authors propose a variant of SSL/TLS in which three connections are established: a direct connection between client and server, and two direct connections between the client and a proxy, and between the proxy and the server. The direct connection can be used for highly confidential data, while the proxy channel can be used for data that doesn't have to remain secret. In [6] the authors propose adding end-to-end encryption to chains of WTLS and TLS connections so that data sent via a proxy remains confidential. However, in both these cases, data can be passed through the proxy without the proxy being able to read it; the proxy can then no longer perform any application-layer jobs it might have (such as virus scanning).

6 Conclusions and future work

We have presented two chaining theorems for secure channels. The theorems are useful because they describe ways in which secure channels might be used, and they allow users of our secure channel specifications to calculate the properties of the overall channel through a proxy very simply. In particular, we have shown that the channels defined in [4] are invariant under chaining through a proxy, provided that the proxy is trustworthy.

In [4] we also present a session property; a session channel guarantees that all the messages received in a connection were sent in a single connection. We also

specify a stream property which guarantees the session property, and that the messages were sent in the same order as that in which they were received. We propose to investigate whether the session and stream properties are invariant under chaining. It seems likely that this is the case (assuming that whenever the proxy receives several messages in a single session he forwards them in a single session, in the same order).

We also intend to investigate the effect of multiple chaining of secure channels. If the chaining is set up as $R_i \to \mathit{Proxy} \to \mathit{Proxy}' \to R_j$, it is not clear what properties the channel through the two different proxies satisfies.

Using the theorems in this paper we could calculate the properties of the overall channel in two different ways: by calculating the resultant channel over the first two connections, then using this result to calculate the result of the overall chain, or by calculating the result of the last two connections first. Because the elevation functions ($\nwarrow_m$ and $\nearrow_m$) are not the same, in most cases these calculations will give different results. For this reason we believe that the overall channel is likely to satisfy the following specification:

$$\downarrow (\nwarrow_m (R_i \to \mathit{Proxy}) \sqcap (\mathit{Proxy} \to \mathit{Proxy}') \sqcap \nearrow_m (\mathit{Proxy}' \to R_j)) .$$

The specifications of the channels to the first proxy and from the last proxy are elevated in the usual way, but there is no elevation on the intermediate channel. It is easy to see how to generalise this result to longer chains.

Acknowledgements

I would like to thank Gavin Lowe for many useful discussions. This work is funded by the US Office of Naval Research.

References

[1] C. Boyd. Security architectures using formal methods. *IEEE Journal on Selected Areas in Communications*, 11(5):694–701, 1993.

[2] P. Broadfoot and G. Lowe. On distributed security transactions that use secure transport protocols. In *Proceedings of the 16th IEEE Computer Security Foundations Workshop*, pages 141–151, 2003.

[3] C. Dilloway. Chaining secure channels. Technical report, Oxford University Computing Laboratory, 2008. Available from http://web.comlab.ox.ac.uk/people/Christopher.Dilloway.html.

[4] C. Dilloway and G. Lowe. Specifying secure channels. In *Proceedings of the 21st IEEE Computer Security Foundations Symposium*, 2008.

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

[6] E. Kwon, Y. Cho, and K. Chae. Integrated transport layer security: End-to-end security model between WTLS and TLS. *Proceedings of the The 15th International Conference on Information Networking*, 2001.

[7] Ueli Maurer and Pierre Schmid. A calculus for secure channel establishment in open networks. In *Computer Security — ESORICS 94*, volume 875 of *Lecture Notes in Computer Science*, pages 175–192. Springer-Verlag, November 1994.

[8] OASIS Security Services Technical Committee. *Assertions and Protocols for the Security Assertion Markup Language (SAML) V2.0*, 2005. Available from http://www.oasis-open.org/committees/security/.

[9] Y. Song, K. Beznosov, and V. Leung. Multiple-channel security architecture and its implementation over SSL. *EURASIP Journal on Wireless Communications and Networking*, 2006(2):78–78, 2006.

[10] Visa International Service Association. *Verified by Visa System Overview External Version 1.0.2*, 2006. Available from https://partnernetwork.visa.com/vpn/global/category.do.

[11] V. Welch, F. Siebenlist, I. Foster, J. Bresnahan, K. Czajkowski, J. Gawor, C. Kesselman, S. Meder, L. Pearlman, and S. Tuecke. Security for grid services. In *Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing*, pages 48–57, 2003.

A Secure channel specifications

A.1 Standard specifications (no proxies)

Definition A.1 (No faking)**.**

$$NF(R_i \to R_j)(tr) \mathrel{\widehat{=}} tr \downarrow \{\!|\, fake.\hat{R}_i.\hat{R}_j \,|\!\} = \langle\rangle\,.$$

Definition A.2 (No-re-ascribing)**.**

$$\begin{aligned} NRA(R_i \to R_j)(tr) \mathrel{\widehat{=}} tr \downarrow \{\!|\, & hijack.A \to A'.B \to B' \mid \\ & A, A' : \hat{R}_i; B, B' : \hat{R}_j \cdot A \neq A' \,|\!\} = \langle\rangle\,. \end{aligned}$$

Definition A.3 (No-honest-re-ascribing)**.**

$$\begin{aligned} NRA^{-}(R_i \to R_j)(tr) \mathrel{\widehat{=}} tr \downarrow \{\!|\, & hijack.A \to A'.B \to B' \mid \\ & A, A' : \hat{R}_i; B, B' : \hat{R}_j \cdot A \neq A' \wedge Honest(A') \,|\!\} = \langle\rangle\,. \end{aligned}$$

Definition A.4 (No-redirecting)**.**

$$\begin{aligned} NR(R_i \to R_j)(tr) \mathrel{\widehat{=}} tr \downarrow \{\!|\, & hijack.A \to A'.B \to B' \mid \\ & A, A' : \hat{R}_i; B, B' : \hat{R}_j \cdot B \neq B' \,|\!\} = \langle\rangle\,. \end{aligned}$$

Definition A.5 (No-honest-redirecting)**.**

$$\begin{aligned} NR^{-}(R_i \to R_j)(tr) \mathrel{\widehat{=}} tr \downarrow \{\!|\, & hijack.A \to A'.B \to B' \mid \\ & A, A' : \hat{R}_i; B, B' : \hat{R}_j \cdot B \neq B' \wedge Honest(B) \,|\!\} = \langle\rangle\,. \end{aligned}$$

A.2 Channel specifications with simple proxies

Definition A.6 (No faking)**.**

$$\begin{aligned} & NF(Proxy_{(R_i,R_j)})(tr) \mathrel{\widehat{=}} \\ & \quad tr \downarrow \{\!|\, fake.\hat{R}_i.\widehat{Proxy}_{(R_i,R_j)}, fake.\widehat{Proxy}_{(R_i,R_j)}.\hat{R}_j \,|\!\} = \langle\rangle\,. \end{aligned}$$

Definition A.7 (No-re-ascribing)**.**

$$\begin{aligned} & NRA(Proxy_{(R_i,R_j)})(tr) \mathrel{\widehat{=}} \\ & \quad tr \downarrow \{\!|\, hijack.A \to A'.P_{(A,B)} \to P_{(A',B')},\ hijack.P_{(A,B)} \to P_{(A',B')}.B \to B' \mid \\ & \qquad A, A' \in \hat{R}_i \wedge B, B' \in \hat{R}_j \wedge P_{(A,B)}, P_{(A',B')} \in \widehat{Proxy}_{(R_i,R_j)} \wedge \\ & \qquad A \neq A' \,|\!\} = \langle\rangle\,. \end{aligned}$$

Definition A.8 (No-honest-re-ascribing)**.**

$$\begin{aligned} & NRA^{-}(Proxy_{(R_i,R_j)})(tr) \mathrel{\widehat{=}} \\ & \quad tr \downarrow \{\!|\, hijack.A \to A'.P_{(A,B)} \to P_{(A',B')},\ hijack.P_{(A,B)} \to P_{(A',B')}.B \to B' \mid \\ & \qquad A, A' \in \hat{R}_i \wedge B, B' \in \hat{R}_j \wedge P_{(A,B)}, P_{(A',B')} \in \widehat{Proxy}_{(R_i,R_j)} \wedge \\ & \qquad A \neq A' \wedge Honest(A') \,|\!\} = \langle\rangle\,. \end{aligned}$$

Definition A.9 (No-redirecting)**.**

$$\begin{aligned} & NR(Proxy_{(R_i,R_j)})(tr) \mathrel{\widehat{=}} \\ & \quad tr \downarrow \{\!|\, hijack.A \to A'.P_{(A,B)} \to P_{(A',B')},\ hijack.P_{(A,B)} \to P_{(A',B')}.B \to B' \mid \\ & \qquad A, A' \in \hat{R}_i \wedge B, B' \in \hat{R}_j \wedge P_{(A,B)}, P_{(A',B')} \in \widehat{Proxy}_{(R_i,R_j)} \wedge \\ & \qquad B \neq B' \,|\!\} = \langle\rangle\,. \end{aligned}$$

Definition A.10 (No-honest-redirecting)**.**

$$NR^{-}(Proxy_{(R_i,R_j)})(tr) \mathrel{\widehat{=}}$$
$$tr \downarrow \{|\, hijack.A \rightarrow A'.P_{(A,B)} \rightarrow P_{(A',B')},\ hijack.P_{(A,B)} \rightarrow P_{(A',B')}.B \rightarrow B' \mid$$
$$A, A' \in \hat{R}_i \wedge B, B' \in \hat{R}_j \wedge P_{(A,B)}, P_{(A',B')} \in \widehat{Proxy}_{(R_i,R_j)} \wedge$$
$$B \neq B' \wedge Honest(B) \,|\} = \langle\rangle\,.$$

A.3 Channel specifications with multiplexing proxies

Definition A.11 (No faking)**.**

$$NF(Proxy(R_i \rightarrow R_j))(tr) \mathrel{\widehat{=}}$$
$$tr \upharpoonright \{|\, fake.A.P.c_A.\langle m, B\rangle,\ fake.P.c_P.B.\langle A, m\rangle \mid$$
$$A \in \hat{R}_i \wedge P \in \widehat{Proxy} \wedge B \in \hat{R}_j \wedge$$
$$c_A, c_P \in Connection \wedge m \in Message_{App} \,|\} = \langle\rangle\,.$$

Definition A.12 (No-re-ascribing)**.**

$$NRA(Proxy(R_i \rightarrow R_j))(tr) \mathrel{\widehat{=}}$$
$$tr \upharpoonright \{|\, hijack.A \rightarrow A'.P \rightarrow P'.c_{P'}.\langle m, B\rangle \mid$$
$$A, A' \in \hat{R}_i \wedge P, P' \in \widehat{Proxy} \wedge B \in \hat{R}_j \wedge c_{P'} \in Connection \wedge$$
$$m \in Message_{App} \wedge A \neq A' \,|\} = \langle\rangle\,.$$

Definition A.13 (No-honest-re-ascribing)**.**

$$NRA^{-}(Proxy(R_i \rightarrow R_j))(tr) \mathrel{\widehat{=}}$$
$$tr \upharpoonright \{|\, hijack.A \rightarrow A'.P \rightarrow P'.c_{P'}.\langle m, B\rangle \mid$$
$$A, A' \in \hat{R}_i \wedge P, P' \in \widehat{Proxy} \wedge B \in \hat{R}_j \wedge c_{P'} \in Connection \wedge$$
$$m \in Message_{App} \wedge A \neq A' \wedge Honest(A') \,|\} = \langle\rangle\,.$$

Definition A.14 (No-redirecting)**.**

$$NR(Proxy(R_i \rightarrow R_j))(tr) \mathrel{\widehat{=}}$$
$$tr \upharpoonright \{|\, hijack.P \rightarrow P'.B \rightarrow B'.c_{B'}.\langle A, m\rangle \mid$$
$$A \in \hat{R}_i \wedge P, P' \in \widehat{Proxy} \wedge B, B' \in \hat{R}_j \wedge c_{B'} \in Connection \wedge$$
$$m \in Message_{App} \wedge B \neq B' \,|\} = \langle\rangle\,.$$

Definition A.15 (No-honest-redirecting)**.**

$$NR^{-}(Proxy(R_i \rightarrow R_j))(tr) \mathrel{\widehat{=}}$$
$$tr \upharpoonright \{|\, hijack.P \rightarrow P'.B \rightarrow B'.c_{B'}.\langle A, m\rangle \mid$$
$$A \in \hat{R}_i \wedge P, P' \in \widehat{Proxy} \wedge B, B' \in \hat{R}_j \wedge c_{B'} \in Connection \wedge$$
$$m \in Message_{App} \wedge B \neq B' \wedge Honest(B) \,|\} = \langle\rangle\,.$$

B Chaining theorem result tables

B.1 Simple proxies

See Figure 2.

B.2 Multiplexing proxies

See Figure 3.

Channel to proxy				Channel from proxy				Resultant channel			
C	NF	NRA	NR	C	NF	NRA	NR	C	NF	NRA	NR
C	NF	NRA	NR	C	NF	NRA	NR^-	C	NF	NRA	NR
C	NF	NRA	NR	C	NF	NRA^-	NR	C	NF	NRA^-	NR
C	NF	NRA	NR	C		NRA	NR^-	C		NRA	NR^-
C	NF	NRA	NR	C	NF	NRA^-	NR^-	C	NF	NRA^-	NR
C	NF	NRA	NR		NF	NRA^-	NR		NF	NRA^-	NR
C	NF	NRA	NR	C		NRA^-	NR^-	C		NRA^-	NR^-
C	NF	NRA	NR		NF	NRA^-	NR^-		NF	NRA^-	NR
C	NF	NRA	NR	C			NR^-	C			NR^-
C	NF	NRA	NR		NF	NRA^-	NR^-		NF	NRA^-	NR
C	NF	NRA	NR			$\perp$				$\perp$	
C	NF	NRA	NR^-	C	NF	NRA	NR	C	NF	NRA	NR
C	NF	NRA	NR^-	C	NF	NRA	NR^-	C	NF	NRA	NR
C	NF	NRA	NR^-	C	NF	NRA^-	NR	C	NF	NRA^-	NR
C	NF	NRA	NR^-	C		NRA	NR^-	C		NRA	NR^-
C	NF	NRA	NR^-	C	NF	NRA^-	NR^-	C	NF	NRA^-	NR
C	NF	NRA	NR^-		NF	NRA^-	NR		NF	NRA^-	NR
C	NF	NRA	NR^-	C		NRA^-	NR^-	C		NRA^-	NR^-
C	NF	NRA	NR^-		NF	NRA^-	NR^-		NF	NRA^-	NR
C	NF	NRA	NR^-	C			NR^-	C			NR^-
C	NF	NRA	NR^-		NF	NRA^-			NF	NRA^-	NR
C	NF	NRA	NR^-			$\perp$				$\perp$	
C	NF	NRA^-	NR	C	NF	NRA	NR	C	NF	NRA	NR
C	NF	NRA^-	NR	C	NF	NRA	NR^-	C	NF	NRA	NR
C	NF	NRA^-	NR	C	NF	NRA^-	NR	C	NF	NRA^-	NR
C	NF	NRA^-	NR	C		NRA	NR^-	C		NRA	NR^-
C	NF	NRA^-	NR	C	NF	NRA^-	NR^-	C	NF	NRA^-	NR
C	NF	NRA^-	NR		NF	NRA^-	NR		NF	NRA^-	NR
C	NF	NRA^-	NR	C		NRA^-	NR^-	C		NRA^-	NR^-
C	NF	NRA^-	NR		NF	NRA^-	NR		NF	NRA^-	NR
C	NF	NRA^-	NR			$\perp$				$\perp$	
C		NRA	NR^-	C	NF	NRA	NR	C		NRA	NR^-
C		NRA	NR^-	C	NF	NRA	NR^-	C		NRA	NR^-
C		NRA	NR^-	C	NF	NRA^-	NR	C		NRA^-	NR^-
C		NRA	NR^-	C		NRA	NR^-	C		NRA	NR^-
C		NRA	NR^-	C	NF	NRA^-	NR^-	C		NRA^-	NR^-
C		NRA	NR^-		NF	NRA^-	NR			$\perp$	
C		NRA	NR^-	C		NRA^-	NR^-	C		NRA^-	NR^-
C		NRA	NR^-		NF	NRA^-	NR^-			$\perp$	

Channel to proxy				Channel from proxy				Resultant channel			
C		NRA	NR^-	C			NR^-	C			NR^-
C		NRA	NR^-		NF	NRA^-				$\perp$	
C		NRA	NR^-			$\perp$				$\perp$	
C	NF	NRA^-	NR^-	C	NF	NRA	NR	C	NF	NRA	NR
C	NF	NRA^-	NR^-	C	NF	NRA	NR^-	C	NF	NRA	NR
C	NF	NRA^-	NR^-	C	NF	NRA^-	NR	C	NF	NRA^-	NR
C	NF	NRA^-	NR^-	C		NRA	NR^-	C		NRA	NR^-
C	NF	NRA^-	NR^-	C	NF	NRA^-	NR^-	C	NF	NRA^-	NR
C	NF	NRA^-	NR^-		NF	NRA^-	NR		NF	NRA^-	NR
C	NF	NRA^-	NR^-	C			NR^-	C			NR^-
C	NF	NRA^-	NR^-		NF	NRA^-			NF	NRA^-	NR
C	NF	NRA^-	NR^-			$\perp$				$\perp$	
	NF	NRA^-	NR	C	NF	NRA	NR		NF	NRA^-	NR
	NF	NRA^-	NR	C	NF	NRA	NR^-		NF	NRA^-	NR
	NF	NRA^-	NR	C	NF	NRA^-	NR		NF	NRA^-	NR
	NF	NRA^-	NR	C		NRA	NR^-			$\perp$	
	NF	NRA^-	NR	C	NF	NRA^-	NR^-		NF	NRA^-	NR
	NF	NRA^-	NR		NF	NRA^-	NR		NF	NRA^-	NR
	NF	NRA^-	NR	C			NR^-			$\perp$	
	NF	NRA^-	NR		NF	NRA^-			NF	NRA^-	NR
	NF	NRA^-	NR			$\perp$				$\perp$	
C		NRA	NR^-	C	NF	NRA	NR	C		NRA	NR^-
C		NRA	NR^-	C	NF	NRA	NR^-	C		NRA	NR^-
C		NRA	NR^-	C	NF	NRA^-	NR	C		NRA^-	NR^-
C		NRA	NR^-	C		NRA	NR^-	C		NRA	NR^-
C		NRA	NR^-	C	NF	NRA^-	NR^-	C		NRA^-	NR^-
C		NRA	NR^-		NF	NRA^-	NR			$\perp$	
C		NRA	NR^-	C		NRA^-	NR^-	C		NRA^-	NR^-
C		NRA	NR^-		NF	NRA^-	NR		NF	NRA^-	NR
C		NRA	NR^-	C	NF	NRA^-	NR	C	NF	NRA^-	NR
C		NRA	NR^-	C		NRA	NR^-	C		NRA	NR^-
C		NRA	NR^-		NF	NRA^-	NR^-		NF	NRA^-	NR
C		NRA	NR^-			$\perp$				$\perp$	

Channel to proxy				Channel from proxy				Resultant channel			
	NF	NRA^-	NR^-	C	NF	NRA^-	NR^-		NF	NRA^-	NR
	NF	NRA^-	NR^-		NF	NRA^-	NR		NF	NRA^-	NR
	NF	NRA^-	NR^-	C		NRA^-	NR^-			$\perp$	
	NF	NRA^-	NR^-		NF	NRA^-	NR^-		NF	NRA^-	NR
	NF	NRA^-	NR^-	C			NR^-			$\perp$	
	NF	NRA^-	NR^-		NF	NRA^-				$\perp$	
		$\perp$		C	NF	NRA	NR			$\perp$	
		$\perp$		C	NF	NRA	NR^-			$\perp$	
		$\perp$		C	NF	NRA^-	NR			$\perp$	
		$\perp$		C		NRA^-	NR^-			$\perp$	
		$\perp$		C	NF	NRA^-	NR^-			$\perp$	
		$\perp$			NF	NRA^-	NR			$\perp$	
		$\perp$		C		NRA^-	NR^-			$\perp$	
		$\perp$			NF	NRA^-	NR^-			$\perp$	
		$\perp$		C			NR^-			$\perp$	
		$\perp$			NF	NRA^-				$\perp$	
		$\perp$				$\perp$				$\perp$	

Figure 2: The table of resultant channels through a simple proxy.

Channel to proxy				Channel from proxy				Resultant channel			
C	NF	NRA	NR	C	NF	NRA	NR	C	NF	NRA	NR
C	NF	NRA	NR	C	NF	NRA	NR^-	C	NF	NRA	NR^-
C	NF	NRA	NR	C	NF	NRA^-	NR	C	NF	NRA	NR
C	NF	NRA	NR	C		NRA	NR^-	C		NRA	NR^-
C	NF	NRA	NR	C	NF	NRA^-	NR^-	C	NF	NRA	NR^-
C	NF	NRA	NR		NF	NRA^-	NR		NF	NRA^-	NR
C	NF	NRA	NR	C		NRA^-	NR^-	C		NRA	NR^-
C	NF	NRA	NR		NF	NRA^-	NR^-		NF	NRA^-	NR^-
C	NF	NRA	NR	C			NR^-	C		NRA	NR^-
C	NF	NRA	NR		NF	NRA^-			NF	NRA^-	
C	NF	NRA	NR			$\perp$				$\perp$	
C	NF	NRA	NR^-	C	NF	NRA	NR	C	NF	NRA	NR
C	NF	NRA	NR^-	C	NF	NRA	NR^-	C	NF	NRA	NR^-
C	NF	NRA	NR^-	C	NF	NRA^-	NR	C	NF	NRA	NR
C	NF	NRA	NR^-	C		NRA	NR^-	C		NRA	NR^-
C	NF	NRA	NR^-	C	NF	NRA^-	NR^-	C	NF	NRA	NR^-
C	NF	NRA	NR^-		NF	NRA^-	NR		NF	NRA^-	NR
C	NF	NRA	NR^-	C		NRA^-	NR^-	C		NRA	NR^-
C	NF	NRA	NR^-		NF	NRA^-	NR^-		NF	NRA^-	NR^-
C	NF	NRA	NR^-	C			NR^-	C		NRA	NR^-
C	NF	NRA	NR^-		NF	NRA^-			NF	NRA^-	
C	NF	NRA	NR^-			$\perp$				$\perp$	
C	NF	NRA^-	NR	C	NF	NRA	NR	C	NF	NRA^-	NR
C	NF	NRA^-	NR	C	NF	NRA	NR^-	C	NF	NRA^-	NR^-
C	NF	NRA^-	NR	C	NF	NRA^-	NR	C	NF	NRA^-	NR
C	NF	NRA^-	NR	C		NRA	NR^-	C		NRA^-	NR^-
C	NF	NRA^-	NR	C	NF	NRA^-	NR^-	C	NF	NRA^-	NR^-
C	NF	NRA^-	NR		NF	NRA^-	NR		NF	NRA^-	NR
C	NF	NRA^-	NR	C		NRA^-	NR^-	C		NRA^-	NR^-
C	NF	NRA^-	NR		NF	NRA^-	NR^-		NF	NRA^-	NR^-
C	NF	NRA^-	NR			$\perp$				$\perp$	
C		NRA	NR^-	C	NF	NRA	NR	C		NRA	NR^-
C		NRA	NR^-	C	NF	NRA	NR^-	C		NRA	NR^-
C		NRA	NR^-	C	NF	NRA^-	NR	C		NRA	NR^-
C		NRA	NR^-	C		NRA	NR^-	C		NRA	NR^-
C		NRA	NR^-	C	NF	NRA^-	NR^-	C		NRA	NR^-
C		NRA	NR^-		NF	NRA^-	NR			$\perp$	
C		NRA	NR^-	C		NRA^-	NR^-	C		NRA	NR^-
C		NRA	NR^-		NF	NRA^-	NR^-			$\perp$	

Channel to proxy				Channel from proxy				Resultant channel			
C		NRA	NR^-	C			NR^-	C		NRA	NR^-
C		NRA	NR^-		NF	NRA^-				$\perp$	
C		NRA	NR^-			$\perp$				$\perp$	
C	NF	NRA^-	NR^-	C	NF	NRA	NR	C	NF	NRA^-	NR
C	NF	NRA^-	NR^-	C	NF	NRA	NR^-	C	NF	NRA^-	NR^-
C	NF	NRA^-	NR^-	C	NF	NRA^-	NR	C	NF	NRA^-	NR
C	NF	NRA^-	NR^-	C		NRA	NR^-	C		NRA^-	NR^-
C	NF	NRA^-	NR^-	C	NF	NRA^-	NR^-	C	NF	NRA^-	NR^-
C	NF	NRA^-	NR^-		NF	NRA^-	NR		NF	NRA^-	NR
C	NF	NRA^-	NR^-	C		NRA^-	NR^-	C		NRA^-	NR^-
C	NF	NRA^-	NR^-		NF	NRA^-	NR^-		NF	NRA^-	NR^-
C	NF	NRA^-	NR^-	C			NR^-	C		NRA^-	NR^-
C	NF	NRA^-	NR^-		NF	NRA^-			NF	NRA^-	
C	NF	NRA^-	NR^-			$\perp$				$\perp$	
C		NRA^-	NR^-	C	NF	NRA	NR	C		NRA^-	NR^-
C		NRA^-	NR^-	C	NF	NRA	NR^-	C		NRA^-	NR^-
C		NRA^-	NR^-	C	NF	NRA^-	NR	C		NRA^-	NR^-
C		NRA^-	NR^-	C		NRA	NR^-	C		NRA^-	NR^-
C		NRA^-	NR^-	C	NF	NRA^-	NR^-	C		NRA^-	NR^-
C		NRA^-	NR^-		NF	NRA^-	NR			$\perp$	
C		NRA^-	NR^-	C		NRA^-	NR^-	C		NRA^-	NR^-
C		NRA^-	NR^-		NF	NRA^-	NR^-			$\perp$	
C		NRA^-	NR^-	C		NRA^-	NR^-	C		NRA^-	NR^-
C		NRA^-	NR^-		NF	NRA^-	NR^-			$\perp$	
C		NRA^-	NR^-			$\perp$				$\perp$	
C		NRA^-	NR^-	C		NRA^-	NR^-	C		NRA^-	NR^-
C		NRA^-	NR^-		NF	NRA^-	NR^-			$\perp$	
C		NRA^-	NR^-	C		NRA^-	NR^-	C		NRA^-	NR^-
C		NRA^-	NR^-		NF	NRA^-	NR^-			$\perp$	
C		NRA^-	NR^-			$\perp$				$\perp$	
NF		NRA^-	NR^-	C	NF	NRA	NR	NF		NRA^-	NR^-
NF		NRA^-	NR^-	C	NF	NRA	NR^-	NF		NRA^-	NR^-
NF		NRA^-	NR^-	C	NF	NRA^-	NR	NF		NRA^-	NR
NF		NRA^-	NR^-	C		NRA	NR^-	NF		NRA^-	NR^-

Channel to proxy				Channel from proxy				Resultant channel			
	NF	NRA^-	NR^-	C	NF	NRA^-	NR^-		NF	NRA^-	NR^-
	NF	NRA^-	NR^-		NF	NRA^-	NR		NF	NRA^-	NR
	NF	NRA^-	NR^-	C		NRA^-	NR^-			$\perp$	
	NF	NRA^-	NR^-		NF	NRA^-	NR^-		NF	NRA^-	NR^-
	NF	NRA^-	NR^-	C			NR^-			$\perp$	
	NF	NRA^-	NR^-		NF	NRA^-			NF	NRA^-	
	NF	NRA^-	NR^-			$\perp$				$\perp$	
C			NR^-	C	NF	NRA	NR	C			NR^-
C			NR^-	C	NF	NRA	NR^-	C			NR^-
C			NR^-	C	NF	NRA^-	NR	C			NR^-
C			NR^-	C		NRA	NR^-	C			NR^-
C			NR^-	C	NF	NRA^-	NR^-	C			NR^-
C			NR^-		NF	NRA^-	NR			$\perp$	
C			NR^-	C		NRA^-	NR^-	C			NR^-
C			NR^-		NF	NRA^-	NR^-			$\perp$	
C			NR^-	C			NR^-	C			NR^-
C			NR^-		NF	NRA^-				$\perp$	
C			NR^-			$\perp$				$\perp$	
	NF	NRA^-		C	NF	NRA	NR		NF	NRA^-	NR
	NF	NRA^-		C	NF	NRA	NR^-		NF	NRA^-	NR^-
	NF	NRA^-		C	NF	NRA^-	NR		NF	NRA^-	NR
	NF	NRA^-		C		NRA^-	NR^-		NF	NRA^-	NR^-
	NF	NRA^-		C	NF	NRA^-	NR^-		NF	NRA^-	NR^-
	NF	NRA^-			NF	NRA^-	NR		NF	NRA^-	NR
	NF	NRA^-		C		NRA^-	NR^-		NF	NRA^-	NR^-
	NF	NRA^-			NF	NRA^-	NR^-		NF	NRA^-	NR^-
	NF	NRA^-				$\perp$				$\perp$	
		$\perp$		C	NF	NRA	NR			$\perp$	
		$\perp$		C	NF	NRA	NR^-			$\perp$	
		$\perp$		C	NF	NRA^-	NR			$\perp$	
		$\perp$		C		NRA	NR^-			$\perp$	
		$\perp$		C	NF	NRA^-	NR^-			$\perp$	
		$\perp$			NF	NRA^-	NR			$\perp$	
		$\perp$		C		NRA^-	NR^-			$\perp$	
		$\perp$			NF	NRA^-	NR^-			$\perp$	
		$\perp$		C			NR^-			$\perp$	
		$\perp$			NF	NRA^-				$\perp$	
		$\perp$				$\perp$				$\perp$	

Figure 3: The table of resultant channels through a multiplexing proxy.