Authentication Tests*

Joshua D. GUTTMAN F. Javier THAYER Fábrega

The MITRE Corporation
{guttman,jt}@mitre.org

Abstract

Suppose a principal in a cryptographic protocol creates and transmits a message containing a new value v, which it later receives back in cryptographically altered form. It can conclude that some principal possessing the relevant key has transformed the message containing v. In some circumstances, this must be a regular participant of the protocol, not the penetrator.

An inference of this kind is an authentication test. *We introduce two main kinds of authentication test. An outgoing test is one in which the new value v is transmitted in encrypted form, and only a regular participant can extract it from that form. An incoming test is one in which v is received back in encrypted form, and only a regular participant can put it in that form. We combine these two tests with a supplementary idea, the unsolicited test, and a related method for checking that certain values remain secret. Together, they determine what authentication properties are achieved by a wide range of cryptographic protocols.*

In this paper we introduce authentication tests and illustrate their power, giving new and straightforward proofs of security goals for several protocols. We also illustrate how to use the authentication tests as a heuristic for finding attacks against incorrect protocols. Finally, we suggest a protocol design process.

We express these ideas in the strand space formalism [21], and prove them correct elsewhere [8].

1 Introduction

A major reason why cryptographic protocol analysis is hard is that the attacker has so many choices. He may apply a repertory of actions in any order to any message he observes, and he may submit the results in place of any legitimate message. In addition, the attacker may initiate new sessions of the protocol, or await sessions initiated by regular participants [6]. Consequently, even though cryptographic protocols are simple finite state activities in the absence of an attacker, the analysis of possible attacks is not necessarily decidable; indeed, even if the protocols are restricted so that the problem is decidable, it may not be tractable [2]. However, everything the penetrator can accomplish can still be achieved if his actions are restricted to a particular order. Although this "normal form lemma" is not new [4, 2], it allows us to justify new methods for establishing authentication and secrecy [8]. In the current paper, we will describe but not prove these methods, and illustrate their significance.

An important consequence of the normal form is that, for certain encrypted components of messages, the penetrator cannot apply any non-trivial actions. Those components may be discarded, but if they are delivered to a regular participant, they can only be delivered unaltered. Only regular protocol participants can change these encrypted components in a way that will be accepted by other regular participants. Therefore, this kind of component may be regarded as an *authentication test*: if the contents are later received in transformed form, then it can only be a regular participant, not the penetrator, who has transformed them. In favorable circumstances, it can only be one regular participant, the intended one, who has thereby been authenticated.

We embody these ideas in three authentication results (Section 2.2, Authentication Tests 1–3). These results allow us to establish many authentication results without any further consideration of the dynamic execution of protocols, which could involve the activity of several principals. Instead, we need only consider the possible behaviors of each principal independently. In Section 3, we illustrate the point by proving the authentication properties of some familiar protocols and identifying counter-examples to some. The protocols we consider are Needham-Schroeder-Lowe [14, 11], Otway-Rees [16], Neuman-Stubblebine [15], and Woo-Lam [22, 23]. It is routine to apply the method to new protocols, whether they use public keys or shared symmetric keys. Apparently, the analysis can be automated in the style

*This work was supported by the National Security Agency through US Army CECOM contract DAAB07-99-C-C201. Appears in *Proceedings, 2000 IEEE Symposium on Security and Privacy*, Oakland CA, May 2000.

of [18].

However, not every protocol can be verified using these methods. In particular, for the authentication theorems to apply, the protocol must not allow the authentication test components to be proper sub-messages of other messages manipulated by the regular participants. We end (Section 4) by suggesting a design process that structures protocols around the authentication tests that show them to be comprehensively correct.

In this paper we emphasize the authentication tests themselves (Section 2.2) and the ease of applying them (Section 3). The proofs justifying the authentication tests are more complicated, and we have segregated them elsewhere [8]. The authentication tests are like the interface to a module; the implementation internal to the module is complex, but the interface is simple, so one can use its services without worrying about the internals. For some purposes it would be helpful to enlarge the interface. There are additional services, or ways of drawing conclusions about authentication protocols, that the proof methods of [8] can offer. One addition would make explicit the order in which events have occurred; this gives a convenient way to reason about whether a key has been generated recently. Another addition would model explicitly the way a key may be generated by hashing other values (as is used e.g. in the SSL and TLS protocols [5]). However, the authentication tests currently exported in Section 2 already apply to a wide range of protocols, and give highly intuitive explanations for why they are right, or where they go wrong.

1.1 Strand Spaces

We very briefly summarize the ideas behind the strand space model [21]; see also Appendix A.

A is the set of messages that can be sent between principals. We call elements of A *terms*. A is freely generated from two disjoint sets, T (representing texts such as nonces or names) and K (representing keys) by means of concatenation and encryption. The concatenation of terms g and h is denoted $g\,h$, and the encryption of h using key K is denoted $\{|h|\}_K$. (See Appendix A.3.)

A term t is a *subterm* of another term t', written $t \sqsubset t'$, if starting with t we can reach t' by repeatedly concatenating with arbitrary terms and encrypting with arbitrary keys. Hence, $K \not\sqsubset \{|t|\}_K$, except in case $K \sqsubset t$. The subterms of t are the values that are uttered when t is sent; in $\{|t|\}_K$, K is not uttered but used. (See Definition A.7.)

A *strand* is a sequence of message transmissions and receptions, where transmission of a term t is represented as $+t$ and reception of term t is represented as $-t$. A strand element is called a *node*. If s is a strand, $\langle s, i\rangle$ is the i^{th} node on s. The relation $n \Rightarrow n'$ holds between nodes n and n' if $n = \langle s, i\rangle$ and $n' = \langle s, i+1\rangle$. Hence, $n \Rightarrow^+ n'$ means that $n = \langle s, i\rangle$ and $n' = \langle s, j\rangle$ for some $j > i$. The relation $n \rightarrow n'$ represents inter-strand communication; it means that $\mathrm{term}(n_1) = +t$ and node $\mathrm{term}(n_2) = -t$.

A *strand space* Σ is a set of strands. The two relations $\Rightarrow$ and $\rightarrow$ jointly impose a graph structure on the nodes of Σ. The vertices of this graph are the nodes, and the edges are the union of $\Rightarrow$ and $\rightarrow$.

We say that a term t *originates* at a node $n = \langle s, i\rangle$ if the sign of n is positive; $t \sqsubset \mathrm{term}(n)$; and $t \not\sqsubset \mathrm{term}(\langle s, i'\rangle)$ for every $i' < i$. Thus, n represents a message transmission that includes t, and it is the first node in s including t. If a value originates on only one node in the strand space, we call it *uniquely originating*; uniquely originating values are desirable as nonces and session keys.

A bundle is a causally well-founded collection of nodes and arrows of both kinds. In a bundle, when a strand receives a message m, there is a unique node transmitting m from which the message was immediately received. By contrast, when a strand transmits a message m, many strands (or none) may immediately receive m. (See Definition A.3.) The height of a strand in a bundle is the number of nodes on the strand that are in the bundle. Authentication theorems generally assert that a strand has at least a given height in some bundle, meaning that the principal must have engaged in at least that many steps of its run.

A strand represents the local view of a participant in a run of a protocol. For a legitimate participant, it represents the messages that participant would send or receive as part of one particular run of his side of the protocol. We call a strand representing a legitimate participant a *regular* strand. For the penetrator, the strand represents an atomic deduction. More complex actions can be formed by connecting several penetrator strands. While regular principals are represented only by what they say and hear, the behavior of the penetrator is represented more explicitly, because the values he deduces are treated as if they had been said publicly.

We partition penetrator strands according to the operations they exemplify. E-strands encrypt when given a key and a plaintext; D-strands decrypt when given a decryption key and matching ciphertext; C-strands and S-strands concatenate and separate terms, respectively; K-strands emit keys from a set of known keys; and M-strands emit known atomic texts or guesses. (See Definition A.9.)

1.2 New Components

When a node transmits or receives a concatenated message, the penetrator—using C-strands and S-strands—has full power over how the parts are concatenated together. Thus, the important units for protocol correctness are what we call the *components*. A term t_0 is a component of t if $t_0 \sqsubset t$, t_0 is not a concatenated term, and every $t_1 \neq t_0$ such that $t_0 \sqsubset t_1 \sqsubset t$ is a concatenated term. Components are ei-

ther atomic values or encryptions. (See Definition A.8.) For instance, the three components of the concatenated term

$$B \, \{|N_a \, K \, \{|K \, N_b|\}_{K_B}|\}_{K_A} \, N_a$$

are B, $\{|N_a \, K \, \{|K \, N_b|\}_{K_B}|\}_{K_A}$, and N_a. We say t is a component of a node n if t is a component of term(n).

A term t is *new* at $n = \langle s, i \rangle$ if t is a component of term(n), but t is not a component of node $\langle s, j \rangle$ for every $j < i$ (Definition A.8). A component is new even if it has occurred earlier as a nested subterm of some larger component $\cdots \{|\cdots t \cdots|\}_K \cdots$.

When a component occurs new on a regular node, then the principal executing that strand has done some cryptographic work to produce the new component. The idea of emphasizing components and the regular nodes at which they occur new is due to Song [18].

2 A Method for Authentication

In this section we describe our method for establishing authentication results. We first show how to establish whether keys are accessible to the penetrator or not (Section 2.1). We then introduce the notion of a *transformed edge*, in which a value is sent out and later received in a new component, and the notion of a *transforming edge*, in which a value is received and later sent out in a new component. We define two main kinds of authentication tests, and state a theorem about each, showing what other regular nodes must exist in a bundle, if that bundle contains an example of an authentication test. A third, simpler variant of an authentication test is also useful, especially when a key server must authenticate its clients. Proofs are in [8].

2.1 Penetrable Keys and Safe Keys

Given a strand space Σ, we can inductively define the set of keys that may become known to the penetrator. We use the relation $\sqsubset_{\mathfrak{K}}$ defined in Definition A.8; $t_0 \sqsubset_{\mathfrak{K}} t$ means that t_0 occurs as a subterm of t in a position where all encryptions surrounding it use keys $K \in \mathfrak{K}$. Thus, either t can be constructed from t_0 simply by (possibly repeated) concatenation, or else t can be written in the form

$$\cdots \{|\cdots t_0 \cdots|\}_K \cdots$$

where $K \in \mathfrak{K}$ and the dots hide only concatenations and other encryptions with keys in $\mathfrak{K}$. The set $\mathfrak{K}^{-1}$ means the set of inverses of keys in $\mathfrak{K}$.

In the base case of this definition we refer to $\mathsf{K}_{\mathcal{P}}$, which is the set of keys known to the penetrator initially, apart from any protocol activity (Definition A.9).

Definition 2.1 *Let* $\mathsf{P}_0 = \mathsf{K}_{\mathcal{P}}$.

Let $\mathsf{P}_{i+1} = \mathsf{P}_i \cup Y$*, where* $K \in Y$ *if and only if there exists a positive regular node* $n \in \Sigma$ *and a term* t *such that* t *is a new component of* n *and* $K \sqsubset_{\mathsf{P}_i^{-1}} t$.

$\mathsf{P} = \bigcup_i \mathsf{P}_i$.

Thus, either a penetrable key is already penetrated ($\mathsf{K}_{\mathcal{P}}$), or else some regular strand puts it in a form that could allow it to be penetrated, because for each key protecting it, the matching decryption key is already penetrable. The justification for this definition is that any key that becomes available to the penetrator in any bundle is a member of P.

Proposition 2.2 *Let* $\mathcal{C}$ *be a bundle with* $n \in \mathcal{C}$ *and* term$(n) = K$*. Then* $K \in \mathsf{P}$.

P is a conservative approximation, in that it may be larger than the set of keys that the penetrator can really capture. This is the case when the strand that would put the key in danger is not contained in any bundle. We also use the notion of a *safe* key.

Definition 2.3 *Let* S_0 *be the set of keys* K *such that* $K \notin \mathsf{K}_{\mathcal{P}}$ *and there is no positive regular node* $n \in \Sigma$ *and term* t *such that* t *is a new component of* n *and* $K \sqsubset t$.

Let S_{i+1} *be the set of keys* K *such that* $K \notin \mathsf{K}_{\mathcal{P}}$*, and for every positive regular node* $n \in \Sigma$ *and new component* t *of* n*, every occurrence of* K *in* t *lies within an encryption using some key* K_0 *where* $K_0^{-1} \in \mathsf{S}_i$*:*

$$\cdots \{|\cdots K \cdots|\}_{K_0} \cdots$$

$\mathsf{S} = \bigcup_i \mathsf{S}_i$*. When* $K \in \mathsf{S}$*, we say that* K *is* safe *in* Σ.

Evidently, the set of safe keys is disjoint from P. However, there are strand spaces Σ in which there are keys K such that $K \notin \mathsf{P} \cup \mathsf{S}$. In practice, protocol secrecy goals usually amount to showing that keys are in either S_0 or S_1. Larger values of i seem rarely to occur.

Showing that a private key or a long-term symmetric key is in S_0 typically reduces to checking that it is assumed not to be in $\mathsf{K}_{\mathcal{P}}$, because protocols generally avoid emitting terms containing these keys.

Many protocols expect session keys to be generated by a key server, which sends them encrypted in the long-term keys of two principals, and no principal ever re-encrypts a session key under a new key. In a particular session, a session key K may be sent encrypted with long-term keys are not in $\mathsf{K}_{\mathcal{P}}$ (or, if they are asymmetric, their inverses are not in $\mathsf{K}_{\mathcal{P}}$). If the server never re-sends the same session key K in a different session, we can infer that $K \in \mathsf{S}_1$.

There also exist protocols in which the session key is translated, in the sense that it is sent out originally encrypted with one key and is later re-encrypted by another principal under a new key. These protocols can also be correct, although they demand special care. The TMN protocol is a (flawed) example [19]. In the case of a correct protocol of

this form, it would be necessary to show that the session key is in S_i for some $i > 1$.

However, because S_0 and S_1 cover typical protocols, our method for proving secrecy is particularly easy to use. It is also easy to prove that a non-key data value such as a nonce is kept secret in some run of a protocol; one simply shows that every new component containing it protects it with an encryption $\{|h|\}_K$ where $K^{-1} \in \mathsf{S}_i$. Again, typically $i = 0$ or 1.

2.2 Facts about Authentication Tests

Fix some strand space Σ. We identify segments of regular strands that amount to tests. Their presence will guarantee the existence of other regular strands in the bundle.

Definition 2.4 *The edge $n_1 \Rightarrow^+ n_2$ is a* transformed edge *[respectively, a* transforming edge*] for $a \in \mathsf{A}$ if n_1 is positive and n_2 is negative [respectively, n_1 is negative and n_2 is positive], $a \sqsubset term(n_1)$, and there is a new component t_2 of n_2 such that $a \sqsubset t_2$.*

Thus, a transformed edge emits a and later tests for its presence in a new form. A transforming edge receives a and later emits it in a new form. We have chosen to interpret a "form" in which a occurs as a component in which it occurs.

Definition 2.5 *$t = \{|h|\}_K$ is a* test component *for a in n if:*

1. *$a \sqsubset t$ and t is a component of n;*
2. *The term t is not a proper subterm of a component of any regular node $n' \in \Sigma$.*

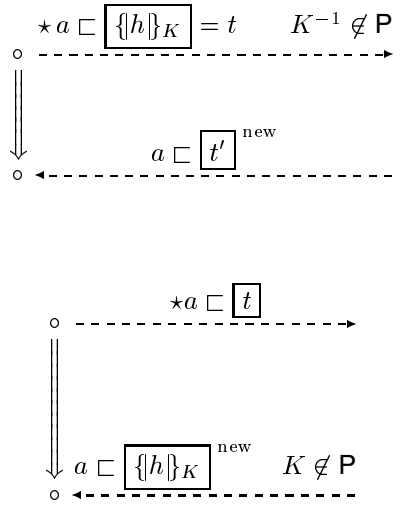
The edge $n_0 \Rightarrow^+ n_1$ is a test *for a if a uniquely originates at n_0 and $n_0 \Rightarrow^+ n_1$ is a transformed edge for a.*

Clause 2 in the definition of test component ensures that the penetrator cannot get any benefit from building a larger term to send to a regular participant, who might then emit some new message of value to the penetrator.

Tests can use their test components in at least two different ways. If the uniquely originating value is sent in encrypted form, and the challenge is to decrypt it, then that is an outgoing test. If it is received back in encrypted form, and the challenge is to produce that encrypted form, then that is an incoming test. These two kinds of test are illustrated in Figure 1.

Definition 2.6 *The edge $n_0 \Rightarrow^+ n_1$ is an* outgoing test *for a in $t = \{|h|\}_K$ if it is a test for a in which: $K^{-1} \notin \mathsf{P}$; a does not occur in any component of n_0 other than t; and t is a test component for a in n_0.*

The edge $n_0 \Rightarrow^+ n_1$ is an incoming test *for a in $t_1 = \{|h|\}_K$ if it is a test for a in which $K \notin \mathsf{P}$ and t_1 is a test component for a in n_1.*



⋆ means a originates uniquely here

$\boxed{t}$ means t is a component of this node

Figure 1. Outgoing and Incoming Tests

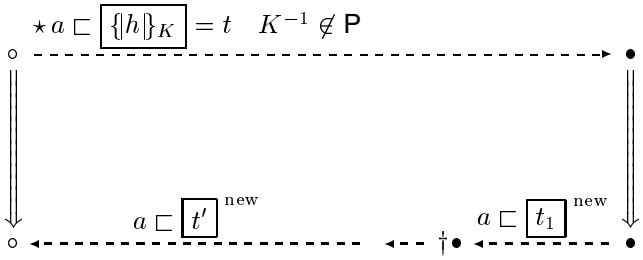
The authentication test results that follow give a powerful method for establishing the authentication goals of protocols.

Authentication Test 1 *Let $\mathcal{C}$ be a bundle with $n' \in \mathcal{C}$, and let $n \Rightarrow^+ n'$ be an outgoing test for a in t.*

1. *There exist regular nodes $m, m' \in \mathcal{C}$ such that t is a component of m and $m \Rightarrow^+ m'$ is a transforming edge for a.*
2. *Suppose in addition that a occurs only in component $t_1 = \{|h_1|\}_{K_1}$ of m', that t_1 is not a proper subterm of any regular component, and that $K_1^{-1} \notin \mathsf{P}$. Then there is a negative regular node with t_1 as a component.*

The meaning of this assertion is illustrated in Figure 2. In this diagram, the two nodes marked $\circ$ represent n and n'. The result assumes that a originates uniquely here (shown by the $\star$), and that the decryption key K^{-1} is safe. The diagram does not represent the assumption that t not be a proper subterm of any regular component, which being non-local is hard to display. The test establishes that $\mathcal{C}$ also contains regular nodes m and m' (marked $\bullet$ at right) with a transforming edge for a. With the assumptions on t_1 given in clause 2, there is also a negative regular node, shown with a $\bullet$ on the bottom line, of which t_1 is a component.

A similar result holds for incoming tests; it can be used to infer authentication results for protocols in which a nonce



• means this regular node must exist

† with additional assumptions on t_1

Figure 2. Authentication Provided by an Outgoing Test

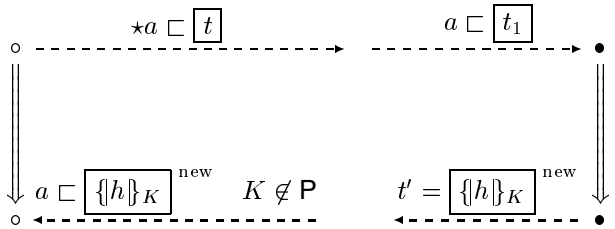


Figure 3. Authentication Provided by an Incoming Test

is emitted in plaintext, for instance as a challenge, and later received in encrypted form.

Authentication Test 2 *Let $\mathcal{C}$ be a bundle with $n' \in \mathcal{C}$, and let $n \Rightarrow^+ n'$ be an incoming test for a in t'. Then there exist regular nodes $m, m' \in \mathcal{C}$ such that t' is a component of m' and $m \Rightarrow^+ m'$ is a transforming edge for a.*

The meaning of this assertion is illustrated in Figure 3 using the same conventions. Although in this paper we will make no use of it, the outgoing and incoming authentication tests also establish an ordering on the nodes, as n occurs before m and m', while n' occurs after. The nodes are ordered $n \prec m \prec m' \prec n'$ in the causal ordering given in Definition A.5. The principal executing n and n' can regard a session key generated at m' as "fresh," because it was created more recently than the beginning of his current run.

The authentication tests are also valid when n and n' are not actually on the same strand, but n is a node known to be in a bundle and to have uniquely originated the test value a, and n' is a node on a different strand that later receives a in transformed form.

The authentication property achieved by an unsolicited test is less informative, but frequently useful, for instance when a key server authenticates its clients.

Definition 2.7 *A negative node n is an* unsolicited test *for $t = \{|h|\}_K$ if t is a test component for any a in n and $K \notin \mathsf{P}$.*

Authentication Test 3 *Let $\mathcal{C}$ be a bundle with $n \in \mathcal{C}$, and let n be an unsolicited test for $t = \{|h|\}_K$. Then there exists a positive regular node $m \in \mathcal{C}$ such that t is a component of m.*

3 Showing Protocol Correctness

In this section we apply the authentication tests of Section 2.2 to several familiar examples. They are the Needham-Schroeder-Lowe public key protocol [14, 11], the Otway-Rees protocol [16, 1, 21], the Neuman-Stubblebine protocol [15, 20], and the Wool-Lam protocol [22, 23]. We do so to illustrate the ease and directness with which these theorems lead to authentication results.

It is remarkably easy to find the outgoing, incoming, and unsolicited tests that provide a protocol's authentication guarantees, assuming that the protocol does not allow its test components to occur in nested contexts. That would violate Clause 2 of the definition of test component (Definition 2.5). The method works for public-key protocols, and for shared symmetric key protocols also.

Outgoing tests provide the authentication guarantees in the Needham-Schroeder-Lowe protocol. In the Otway-Rees protocol, each of the initiator and the responder uses an outgoing test to authenticate a server strand. The server uses an unsolicited test to establish that the initiator and responder have each sent a message. The Neuman-Stubblebine protocol uses a combination of incoming tests and unsolicited tests. It is a two-part protocol. The second part is flawed, both in itself [9] and in undermining the guarantees that part I provide in isolation [20]. We will use the authentication test results to explain both why the first part works in isolation, and also why the addition of the second part undermines its guarantees.

We give a detailed exposition in Section 3.1, so that the reader can see just how our method works. The discussion in Sections 3.2–3.4 is less detailed, as there is no point in repeating the same routine checks.

3.1 Needham-Schroeder-Lowe

Let T_{name} be a distinguished set with $\mathsf{T}_{\text{name}} \subset \mathsf{T}$. In the form we consider, the Needham-Schroeder-Lowe protocol involves two types of regular strands:

1. Initiator strands with trace

$$\langle +\{|N_a\ A|\}_{K_B}, \quad -\{|N_a\ N_b\ B|\}_{K_A}, \quad +\{|N_b|\}_{K_B} \rangle,$$

where $A, B \in \mathsf{T}_{\text{name}}$, $N_a, N_b \in \mathsf{T}$ but $N_a \notin \mathsf{T}_{\text{name}}$. Init$[A, B, N_a, N_b]$ will denote the set of all strands with the trace shown.

2. Complementary responder strands with trace

$$\langle -\{|N_a\ A|\}_{K_B},\quad +\{|N_a\ N_b\ B|\}_{K_A},\quad -\{|N_b|\}_{K_B}\rangle$$

where $A, B \in \mathsf{T}_{\text{name}}$, $N_a, N_b \in \mathsf{T}$ but $N_b \notin \mathsf{T}_{\text{name}}$. Resp$[A, B, N_a, N_b]$ will denote the set of all strands with the trace shown.

Fix a strand space Σ in which all regular strands are of these forms. Correctness depends on the assumption that the "public key of" mapping $f : A \mapsto K_A$ is injective.

We note from the form of the regular strands that, for regular nodes n, $K \in \mathsf{K}$ implies $K \not\sqsubset \text{term}(n)$. Hence, Definition 2.3 yields a result about the secrecy of keys:

Proposition 3.1 *For Σ, $\mathsf{K} = \mathsf{K}_{\mathcal{P}} \cup \mathsf{S}_0$, so $\mathsf{P} = \mathsf{K}_{\mathcal{P}}$.*

Proposition 3.2 *Let $\mathcal{C}$ be a bundle in Σ, and s be a responder strand in* Resp$[A, B, N_a, N_b]$ *with $\mathcal{C}$-height 3. Assume $K_A^{-1} \notin \mathsf{K}_{\mathcal{P}}$. Suppose $N_a \neq N_b$ and N_b is uniquely originating. Then there is an initiator strand $s' \in$* Init$[A, B, N_a, N_b]$ *with $\mathcal{C}$-height 3.*

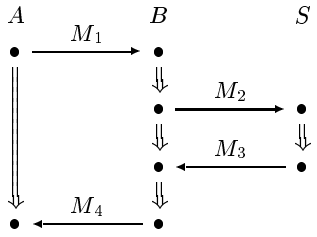
PROOF. We show first that the second and third nodes on s form an outgoing test for N_b. $\{|N_a\ N_b\ B|\}_{K_A}$ is a test component for N_b in $\langle s, 2\rangle$, because it contains N_b, and no regular node has any term of this form as a proper subterm. Checking the assumptions, it follows that $\langle s, 2\rangle \Rightarrow^+ \langle s, 3\rangle$ is an outgoing test for N_b in $\{|N_a\ N_b\ B|\}_{K_A}$.

By Authentication Test 1, there exist regular nodes $m, m' \in \mathcal{C}$ such that $\{|N_a\ N_b\ B|\}_{K_A}$ is a component of m and $m \Rightarrow^+ m'$ is a transforming edge for N_b.

Because m is a negative regular node and $\{|N_a\ N_b\ B|\}_{K_A} = \text{term}(m)$, m is $\langle s', 2\rangle$ for some initiator strand $s' = \text{Init}[A', B', N'_a, N'_b]$. Since $\text{term}(\langle s', 2\rangle) = \{|N_a\ N_b\ B|\}_{K_A}$, we see that $A' = A$, $B' = B$, $N'_a = N_a$, and $N'_b = N_b$. The $\mathcal{C}$-height of s' is 3, because $\langle s', 2\rangle \Rightarrow^+ \langle s', 3\rangle$ is a transforming edge in $\mathcal{C}$. ■

The same proof explains the extent to which the original Needham-Schroeder protocol achieved authentication. In that version, the outgoing test was instead $\{|N_a\ N_b|\}_{K_A}$, lacking B's name. All the reasoning is the same, except it leads only to the conclusion that there is a strand $s' \in$ Init$[A, B', N_a, N_b]$, for some B', with $\mathcal{C}$-height 3. Lowe's attack [10] supplies a scenario in which a responder strand $s \in$ Resp$[A, B, N_a, N_b]$ coexists with an initiator strand $s' \in$ Init$[A, B', N_a, N_b]$ where $B' \neq B$.

We will also prove the initiator's authentication guarantee. The proof is very similar to that of Proposition 3.2, except that it is necessary to use the second part of Authentication Test 1 as well as the first part of it.



$M_1 = M\ A\ B\ \{|N_a\ M\ A\ B|\}_{K_{AS}}$
$M_2 = M\ A\ B\ \{|N_a\ M\ A\ B|\}_{K_{AS}}\ \{|N_b\ M\ A\ B|\}_{K_{BS}}$
$M_3 = M\ \{|N_a\ K_{AB}\ |\}_{K_{AS}}\ \{|N_b\ K_{AB}\ |\}_{K_{BS}}$
$M_4 = M\ \{|N_a\ K_{AB}\ |\}_{K_{AS}}$

Figure 4. Message Exchange in Otway-Rees

Proposition 3.3 *Let $\mathcal{C}$ be a bundle in Σ, and s be an initiator's strand in* Init$[A, B, N_a, N_b]$ *with $\mathcal{C}$-height 3. Assume $K_A^{-1}, K_B^{-1} \notin \mathsf{K}_{\mathcal{P}}$, and suppose that N_a is uniquely originating. Then there is a responder strand $s' \in$* Resp$[A, B, N_a, N_b]$ *with $\mathcal{C}$-height 2.*

PROOF. Observe that the first two nodes of s are an outgoing test for N_a in $\{|N_a\ A|\}_{K_B}$. As in the previous proof, using the first part of Authentication Test 1, it follows that there is a responder strand $s' \in$ Resp$[A, B, N_a, N'_b]$ with $\mathcal{C}$-height 2.

To see that $N'_b = N_b$, we use the second part of Authentication Test 1 to show that there is a negative regular node with component $\{|N_a\ N'_b\ B|\}_{K_A}$. This can only lie on some initiator strand s''. By the form of an initiator strand, N_a originates on s''. Since N_a originates uniquely, $s'' = s$, so $N'_b = N_b$. ■

3.2 The Otway-Rees Protocol

The Otway-Rees protocol (Figure 4) uses long-term symmetric keys shared with a key server to distribute a new session key for a conversation between two clients. The protocol does not establish that the same key is delivered to both A and B [21], only that if either A or B reaches the end of its strand, then the other has submitted the expected matching original request $\{|N_b\ M\ A\ B|\}_{K_{BS}}$ or $\{|N_a\ M\ A\ B|\}_{K_{AS}}$. Also, K is not disclosed, assuming the server chooses a uniquely originating session key K.

3.2.1 Strand Spaces for Otway-Rees

The regular strands are defined to be of the form:

1. Initiator strands in Init$[A, B, N, M, K]$, with trace:

$$\langle +\ M\ A\ B\ \{|N\ M\ A\ B|\}_{K_{AS}}, -\ M\ \{|N\ K|\}_{K_{AS}}\rangle$$

2. Responder strands in $\mathrm{Resp}[A, B, N, M, K, H, H']$ with trace:

$$\langle -\ M\ A\ B\ H,\ +\ M\ A\ B\ H\ \{|N\ M\ A\ B|\}_{K_{BS}},\ -\ M\ H'\ \{|N\ K|\}_{K_{BS}},\ +\ M\ H'\rangle$$

3. Server strands in $\mathrm{Serv}[A, B, N_a, N_b, M, K]$ with trace:

$$\langle -\ M\ A\ B\ \{|N_a\ M\ A\ B|\}_{K_{AS}}\ \{|N_b\ M\ A\ B|\}_{K_{BS}},\ +\ M\ \{|N_a\ K|\}_{K_{AS}}\ \{|N_b\ K|\}_{K_{BS}}\rangle$$

The principal active in $\mathrm{Init}[A, B, N, M, K]$ is A, while the active principal in $\mathrm{Resp}[A, B, N, M, K, **]$ is B.[1] We define LT to be the set of long-term keys, i.e. the range of the injective function K_{AS} for $A \in \mathsf{T}_{\mathrm{name}}$. All long-terms keys are symmetrical: $K \in \mathsf{LT}$ implies $K = K^{-1}$.

We will use three side assumptions.

1. We assume that the responder's nonce originates on that strand, which implies that $\mathrm{Resp}[*, *, N, *, *, H, *] = \emptyset$ if $N \sqsubset H$.

2. We assume that the terms H and H', which are simply forwarded by the responder with no interpretation or processing, contain no proper encrypted subterms. That is, $\{|g|\}_K \sqsubset H$ and $\{|g|\}_K \neq H$ implies

$$\mathrm{Resp}[**, H, *] = \emptyset;$$

and likewise for H'. We prove elsewhere [8] that this assumption does not mask any possible failure of the protocol.[2]

3. We assume that the server generates keys in a reasonable manner, in the sense that $\mathrm{Serv}[**, K] = \emptyset$ unless: $K \notin \mathsf{K}_{\mathcal{P}}$; $K = K^{-1}$; K is uniquely originating; and $K \notin \mathsf{LT}$. It follows from the unique origination assumption that the cardinality $|\mathrm{Serv}[**, K]| \leq 1$ for every K.

[1] We sometimes use an asterisk to indicate a union over a particular argument position, and a double asterisk to indicate a union over all remaining argument positions. Thus, for instance, $\mathrm{Serv}[*, *, *, *, *, K]$ is the set of all server strands emitting the session key K; $\mathrm{Resp}[A, B, N, M, K, **]$ is the set of all responder strands with initiator A, responder B, nonce N, round number M, session key K, and any value of the remaining parameters. We will also abbreviate a form like $\mathrm{Serv}[*, *, *, *, *, K]$ to $\mathrm{Serv}[**, K]$.

[2] In effect, since the responder strands do not depend on the form of H, the penetrator can splice out any value t not meeting the constraint and splice another value t' into its position. Later, after the regular participant has processed the message, t' will be emitted. The penetrator then splices t back into position.

When authentication tests are applied to a protocol using symmetric cryptography and a key server, this trick may always be applied. There is never a problem about whether unconstrained "H-terms" are compatible with the assumption that the test term not be a proper subterm of a regular component.

Let Σ be a strand space satisfying these conditions.

3.2.2 Otway-Rees Authentication

Structurally, Otway-Rees achieves its authentication guarantees in three steps.

1. The long-term keys LT are not uttered by the protocol. Thus, if $K \in \mathsf{LT}$ and $K \notin \mathsf{K}_{\mathcal{P}}$, then $K \in \mathsf{S}_0$. Hence, if the server distributes a session key K' to principals with uncompromised keys, then $K' \in \mathsf{S}_1$.

2. The server strand receives an unsolicited test that authenticates the initial positive node of the initiator and responder.

3. The initiator strand contains an outgoing test for N_a in $\{|N_a\ M\ A\ B|\}_{K_{AS}}$; this authenticates the server strand. Likewise, the responder strand contains an outgoing test for N_b in $\{|N_b\ M\ A\ B|\}_{K_{BS}}$, which authenticates the server strand.

The initiator authenticates the responder only in that it authenticates the server strand, which has authenticated the occurrence of the responder's initial positive node. The situation is symmetrical for the responder authenticating the initiator.

Because $K \not\sqsubset \mathrm{term}(n)$ for long-term keys $K \in \mathsf{LT}$ and regular nodes n, Definition 2.3 immediately entails $\mathsf{LT} \subset \mathsf{S}_0 \cup \mathsf{K}_{\mathcal{P}}$. Because the initiator and responder strands emit no new components in which keys occur, a session key can be compromised only if the server sends it out encrypted with a compromised long-term key. By the unique origination assumption on session keys, if it is sent out under uncompromised long-term keys, then the server will never re-use it with compromised long-term keys. Summarizing this, we have:

Proposition 3.4 $\mathsf{LT} \subset \mathsf{S}_0 \cup \mathsf{K}_{\mathcal{P}}$. *If* $K_{AS}, K_{BS} \notin \mathsf{K}_{\mathcal{P}}$ *and* $\mathrm{Serv}[A, B, *, *, *, K] \neq \emptyset$ *then* $K \in \mathsf{S}_1$.

Turning now to the server's authentication guarantee, we use unsolicited tests.

Proposition 3.5 *Suppose that* $\mathcal{C}$ *is a bundle in* Σ*;* $A \neq B$*;* $K_{AS}, K_{BS} \notin \mathsf{K}_{\mathcal{P}}$*; and* $s \in \mathrm{Serv}[A, B, N_a, N_b, M, *]$ *has* $\mathcal{C}$*-height 1.*

Then there exist $s_i \in \mathrm{Init}[A, B, N_a, M, *]$ *and* $s_r \in \mathrm{Resp}[A, B, N_b, M, **]$ *such that* s_i *has* $\mathcal{C}$*-height 1 and* s_r *has* $\mathcal{C}$*-height 2.*

PROOF. The terms $\{|N_a\ M\ A\ B|\}_{K_{AS}}$ and $\{|N_b\ M\ A\ B|\}_{K_{BS}}$ are unsolicited tests, and therefore (Authentication Test 3) occur on positive regular nodes in $\mathcal{C}$. When $A \neq B$, the latter occurs positively only on a node $\langle s_r, 2\rangle$ where $s_r \in \mathrm{Resp}[A, B, N_b, M, **]$.

As for $\{|N_a\ M\ A\ B|\}_{K_{AS}}$, it may occur positively either on a strand $s_i \in \text{Init}[A, B, N_a, M, *]$ or as H or H' in a strand $s'_r \in \text{Resp}[**, H, *]$ or $\text{Resp}[**, H']$. Let S be the set of all regular nodes in $\mathcal{C}$ having $\{|N_a\ M\ A\ B|\}_{K_{AS}}$ as a component. Since S is non-empty, it has a $\preceq_{\mathcal{C}}$-minimal member n_0 (Proposition A.6). Since neither H nor H' occurs new on a responder strand, n_0 can only be of the form $\langle s_i, 1\rangle$ for $s_i \in \text{Init}[A, B, N_a, M, *]$. ■

If $A = B$, then $\{|N\ M\ A\ B|\}_{K_{AS}} = \{|N\ M\ A\ B|\}_{K_{BS}}$, so the server can no longer be sure that both an initiator strand and a responder strand are present. This is the explanation for the odd attack, attributed to Michael Goldsmith, in which "the responder thinks he wants to talk to himself, but he really doesn't."

$P(B) \to B$: $\quad B\ B\ M\ H$

$B \to P(S)$: $\quad B\ B\ M\ H\ \{|N_b\ M\ B\ B|\}_{K_{BS}}$

$P(B) \to S$: $\quad B\ B\ M\ \{|N_b\ M\ B\ B|\}_{K_{BS}}\ \{|N_b\ M\ B\ B|\}_{K_{BS}}$

which causes a normal server strand, despite the non-existence of any active initiator.

Proposition 3.6 *Suppose that $\mathcal{C}$ is a bundle in Σ; $A \neq B$; $K_{AS} \notin \mathsf{K}_{\mathcal{P}}$; and $s_i \in \text{Init}[A, B, N_a, M, K]$ has $\mathcal{C}$-height 2.*

Then there exists $s \in \text{Serv}[A, B, N_a, *, M, K]$ with $\mathcal{C}$-height 2.

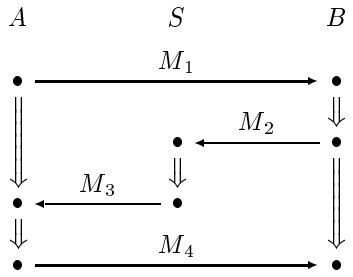
PROOF. $\langle s_i, 1\rangle \Rightarrow^+ \langle s_i, 2\rangle$ is an outgoing test for N_a in $\{|N_a\ M\ A\ B|\}_{K_{AS}}$. Therefore there is a regular transforming edge for N_a (Authentication Test 1). By inspection, this can only lie on a server strand $s \in \text{Serv}[A, B, N_a, *, M, K]$. ■

Proposition 3.7 *Suppose that $\mathcal{C}$ is a bundle in Σ; $A \neq B$; $K_{BS} \notin \mathsf{K}_{\mathcal{P}}$; and $s_r \in \text{Resp}[A, B, N_b, M, K, **]$ has $\mathcal{C}$-height 3.*

Then there exists $s \in \text{Serv}[A, B, *, N_b, M, K]$ with $\mathcal{C}$-height 2.

The proof is similar to that of Proposition 3.6.

These three theorems exhaust the authentication that this protocol actually achieves. Consider, for example, the initiator's guarantee that the responder has been active in a bundle $\mathcal{C}$ containing a strand s_i in $\text{Init}[A, B, N_a, M, K]$. It follows from Proposition 3.6, which establishes that the bundle contains some $s' \in \text{Serv}[A, B, N_a, *, M, K]$, together with Proposition 3.5, which further shows that some $s_r \in \text{Resp}[A, B, *, M, **]$ has $\mathcal{C}$-height 2. Because Proposition 3.5 does not constrain the session keys, the Otway-Rees protocol cannot possibly guarantee that the responder strand (even if completed) will receive the same session key [21].



$M_1 = A\ N_a$

$M_2 = B\ \{|A\ N_a\ t_b|\}_{K_{BS}}\ N_b$

$M_3 = \{|B\ N_a\ K\ t_b|\}_{K_{AS}}\ \{|A\ K\ t_b|\}_{K_{BS}}\ N_b$

$M_4 = \{|A\ K\ t_b|\}_{K_{BS}}\ \{|N_b|\}_K$

Figure 5. Neuman-Stubblebine Part I (Authentication)

3.3 Neuman-Stubblebine

The Neuman-Stubblebine protocol [15] contains two sub-protocols. In the first part the initiator and responder use a key distribution server to authenticate one another and acquire a session key. In the second part the key distribution server is not involved; the initiator re-presents a ticket obtained in a run of part I, and the initiator and responder re-authenticate one another. We will call the first sub-protocol the authentication protocol and the second sub-protocol the re-authentication protocol. In the authentication sub-protocol, a key distribution center generates a session key for an initiator (a network client) and a responder (a network server); the message exchange is shown in Figure 5. This session key is embedded in encrypted form in a re-usable ticket of the form $\{|A\ K\ T|\}_{K_{BS}}$.

Strands of the form shown in the columns labelled A, B, and S in Figure 5 will be called $\text{Init}[A, B, N_a, N_b, t_b, K, H]$, $\text{Resp}[A, B, N_a, N_b, t_b, K]$, and $\text{Serv}[A, B, N_a, N_b, t_b, K]$, respectively.

As in Section 3.2, we define LT to be the set of long-term keys, i.e. the range of the injective function K_{AS} for $A \in \mathsf{T}_{\text{name}}$. All long-terms keys are symmetrical: $K \in \mathsf{LT}$ implies $K = K^{-1}$. We likewise assume that the server generates keys in a reasonable way, meaning that that $\text{Serv}[**, K] = \emptyset$ unless: $K \notin \mathsf{K}_{\mathcal{P}}$; $K = K^{-1}$; K is uniquely originating; and $K \notin \mathsf{LT}$. Because of the unique origination assumption, it follows that the cardinality $|\text{Serv}[**, K]| \leq 1$ for every K.

The initiator's guarantee is simple to establish. Assuming $K_{AS} \notin \mathsf{K}_{\mathcal{P}}$, the edge $M_1 \Rightarrow M_3$ on an initiator

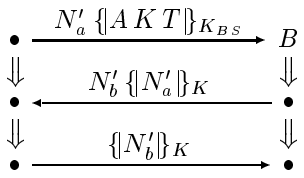


Figure 6. Neuman-Stubblebine, Part II (Re-authentication)

strand Init$[A, B, N_a, N_b, t_b, K, H]$ is an incoming test for N_a in $\{|B\ N_a\ K\ t_b|\}_{K_{AS}}$. It shows there is a server strand $s_s \in$ Serv$[A, B, N_a, *, t_b, K]$. Assuming $K_{BS} \notin \mathsf{K}_\mathcal{P}$, the first node of s_s is an unsolicited test, showing the existence of a responder strand $s_r \in$ Resp$[A, B, N_a, *, t_b, *]$ of $\mathcal{C}$-height 2.

The responder's guarantee is subtler. The overall strategy for showing it, given a strand $s_r \in$ Resp$[A, B, N_a, N_b, t_b, K]$ and assuming $K_{AS}, K_{BS} \notin \mathsf{K}_\mathcal{P}$, is the following:

1. As with Otway-Rees, $\mathsf{LT} \subset \mathsf{S}_0 \cup \mathsf{K}_\mathcal{P}$. So for all K', $K' \in \mathsf{S}_1$ whenever Serv$[A, B, *, *, *, K'] \neq \emptyset$.

2. $\{|A\ K\ t_b|\}_{K_{BS}}$ is an unsolicited test, which can originate only on a regular strand. This can only be a server strand $s_s \in$ Serv$[A, B, *, *, t_b, K]$. By step 1, $K \in \mathsf{S}_1$.

3. $M_2 \Rightarrow M_4$ is an incoming test for N_b in $\{|N_b|\}_K$. Hence, there is a regular transforming edge producing $\{|N_b|\}_K$. This can lie only on the second and third nodes of an initiator strand $s_i \in$ Init$[A', B', N'_a, N_b, t'_b, K, *]$.

4. Since $\langle s_i, 2\rangle$ contains $\{|B'\ N'_a\ K\ t'_b|\}_{K_{A'S}}$ and $K \in \mathsf{S}_1$, it follows that $K^{-1}_{A'S} \notin \mathsf{P}$. Moreover $K^{-1}_{A'S} = K_{A'S}$.

   So $\{|B'\ N'_a\ K\ t'_b|\}_{K_{A'S}}$ is an unsolicited test, which can originate only on a regular strand. This can only be a server strand $s'_s \in$ Serv$[A', B', N'_a, *, t'_b, K]$.

5. Since server strands construct uniquely originating keys, and K originates on both s_s and s'_s, it follows that $s_s = s'_s$. Hence, $A' = A$, $B' = B$, and $t'_b = t_b$. Therefore, $s_i \in$ Init$[A, B, *, N_b, t_b, K, *]$, and this strand has height at least three.

In the re-authentication sub-protocol, the key distribution center no longer needs to be involved; the initiator again presents the same ticket to the responder, as shown in Figure 6. However, in the presence of this additional sub-protocol, step 3 in the responder's guarantee can no longer be completed. There is certainly still a transforming edge producing $\{|N_b|\}_K$, but this edge may lie either on an initiator strand for Part I of the protocol, or on either type of strand for Part II. By contrast, the initiator's guarantee for Part I is unaffected, because we have not added any strand with a transforming edge producing a term of the form $\{|B\ N_a\ K\ t_b|\}_{K_{AS}}$.

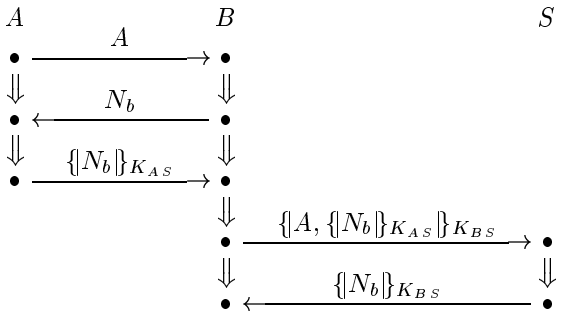


Figure 7. Woo-Lam

3.4 The Woo-Lam Protocol

The Woo-Lam one-way authentication protocol [22] also uses an incoming test, although in a flawed way [23, 3, 7]. It is intended to allow an initiator (client) A to authenticate his presence to a responder (networked service) B, by means of long-term keys shared with a key server. A receives no authenticating information about B. The behavior of the protocol is given in Figure 7.

It is clear from Figure 7 how this is *intended* to work. The $\Rightarrow^+$ edge from B's first transmission of N_b to its final reception of $\{|N_b|\}_{K_{BS}}$ is intended to serve as an incoming test with that term as test component. The server's edge $\{|A, \{|N_b|\}_{K_{AS}}|\}_{K_{BS}} \Rightarrow \{|N_b|\}_{K_{BS}}$ is intended as the corresponding transforming edge. It "authenticates" that the server has found N_b inside A's encrypted message.

Unfortunately, this description is enough to see what is wrong with this protocol. There is another type of transforming edge that produces a term of the same form as the incoming test component. This is the initiator's encrypting edge, in the case in which the initiator is B. Thus, the attacker can wait until B needs to authenticate itself to any responder, and can then execute the attack shown in Figure 8. Woo and Lam state that they assume that a principal can detect when it receives an encrypted unit that it has constructed itself; so perhaps this attack is not entirely "fair." See [3] for additional discussion.

Yet another problem (also discussed in [3]) exists. Even when the server constructs the term $\{|N_b|\}_{K_{BS}}$, this term does not fully determine the parameters to the server strand. A second attack on Woo-Lam exploits this. The attacker starts two sessions with the responder B. In one he purports to be A; in the other he uses some identity C he has some-

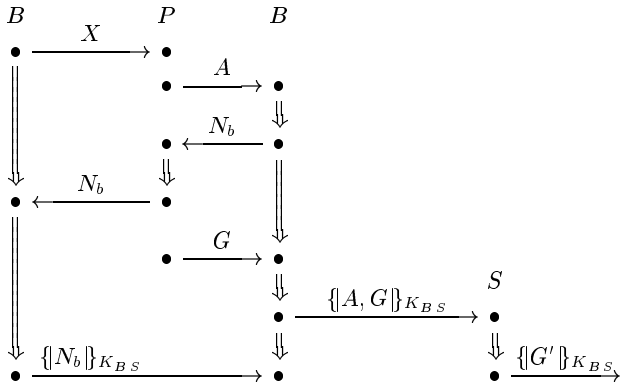


Figure 8. Woo-Lam Infiltrated

how captured, so that $K_{CS} \in \mathsf{K}_{\mathcal{P}}$. He then switches the nonce N_b that B generates, intended to authenticate A, into the session with C, so that B sends $\{|C, \{|N_b|\}_{K_{CS}}|\}_{K_{BS}}$ to the server. The server then generates $\{|N_b|\}_{K_{BS}}$, which is the test component for B's session with A. The attacker then makes this appear to belong to that session. The auxiliary session with C fails to complete.

The Woo-Lam example is included here to illustrate how useful the authentication tests are as a heuristic used to find problems in protocols. They may be used for this purpose even in a case in which some of the official constraints on the authentication test are not satisfied. For instance, in the Woo-Lam protocol, the test component $\{|N_b|\}_{K_{BS}}$ could also occur as a proper subterm of a regular node, namely the message from a responder to the server. However, the authentication tests still model the reasoning of a protocol designer well enough to suggest where failures will lie.

4 Designing a Protocol: A Rational Reconstruction

The outgoing, incoming, and unsolicited tests, and the authentication results that apply to them, suggest a protocol design process. At our level of abstraction, authentication protocol design is largely a matter of selecting authentication tests, and constructing a unique regular transforming edge to satisfy each.[3] We will illustrate this process by an example, a possible rational reconstruction leading to the Needham-Schroeder-Lowe protocol.

It is important to start by deciding the goals to be achieved. Let us assume that we intend to construct a protocol in which the initiator A and responder B each generate a fresh, secret value, N_a and N_b respectively. They want to share these values between themselves without disclosing them to any other party. Each should learn that the other has proceeded far enough in the protocol to have received the values. Perhaps the principals intend to hash the two values together to produce a session key for an encrypted conversation. We will try to accomplish our goals without using excessive messages.

We must also stipulate the cryptographic conditions under which the protocol will operate. In our case, the relevant assumption is that each principal has an asymmetric key pair, and can reliably obtain the other's public key. Perhaps some public key infrastructure is already in place.

From the goal it follows that A can use an authentication test using N_a, while B can use an authentication test using N_b. Given the assumption that the principals hold each other's public keys, this can be an outgoing test. A can use a test component of the form $\{|\cdots N_a \cdots|\}_{K_B}$ assuming K_B^{-1} is uncompromised. Only B will be able to extract N_a from this encrypted form.

By contrast, an incoming test is not suitable. For instance, an incoming component of the form $\{|\cdots N_a \cdots|\}_{K_B^{-1}}$ would ensure that the transforming edge lies on a strand of principal B, but would sacrifice the secrecy of N_a. Similarly, an incoming component of the form $\{|\cdots N_a \cdots|\}_{K_A}$ would preserve secrecy, but would not ensure that the transforming edge lies on a regular strand, much less a strand of principal B. Nested encryption might yield a usable incoming test, but is more computationally demanding and more fragile.

The value A receives back in the outgoing test must be encrypted in a key whose inverse is uncompromised, presumably K_A, to preserve secrecy. In addition, the first term must contain A's name, as otherwise B does not know which public key to use for the return message. Thus, the first steps for A will be of the form

$$+ \{|N_a\ A|\}_{K_B} \Rightarrow - \{|\cdots N_a \cdots|\}_{K_A} \Rightarrow \cdots$$

A similar argument shows that B will use an outgoing test of the form:

$$\cdots \Rightarrow + \{|\cdots N_b \cdots|\}_{K_A} \Rightarrow - \{|\cdots N_b \cdots|\}_{K_B} \Rightarrow \cdots$$

We save a message by observing that B's outgoing message can be combined with A's incoming message. Hence, B's behavior can take the form:

$$\begin{array}{ll} - & \{|N_a\ A|\}_{K_B} \Rightarrow \\ & + \{|N_a\ N_b \cdots|\}_{K_A} \Rightarrow - \{|\cdots N_b \cdots|\}_{K_B} \Rightarrow \cdots \end{array}$$

If we try to be clever, we may guess that the presence of N_a will identify the run to A. In that case, we discard the ellipsis in B's outgoing message. Since there is no need to

[3] Of course, at other levels of abstraction there are other issues, concerning how to negotiate cryptographic algorithms, how to evaluate whether cryptography has been used safely, how to format messages, how to distribute certificates, how to align key streams, and so on, that are not considered at the current level of abstraction.

add anything to the third message or after it, we obtain the Needham-Schroeder protocol:

$$-\{|N_a\ A|\}_{K_B} \Rightarrow +\{|N_a\ N_b|\}_{K_A} \Rightarrow -\{|N_b|\}_{K_B}$$

A more systematic approach is to check whether the values contained in B's outgoing test component suffice to identify a unique initiator strand as the transforming edge for N_b. They do not, because B's identity is not determined. This establishes that we need a correction like Lowe's:

$$-\{|N_a\ A|\}_{K_B} \Rightarrow +\{|N_a\ N_b\ B|\}_{K_A} \Rightarrow -\{|N_b|\}_{K_B}$$

We have now selected the complete message structure for the protocol. We must now check that we have done so correctly. There are five questions that need to be answered:

1. Is the set of penetrable keys P disjoint from the decryption keys for outgoing components, and disjoint from the encryption keys for incoming and unsolicited components?
2. Is any test component a proper subterm of a component of term(n) for any regular node n?
3. Are there ever two types of transforming edge that transform the same outgoing component, or produce the same incoming component?
4. Do the parameters contained in the test components completely determine the data values contained in the desired authentication guarantee?
5. If a data value is intended to remain secret, is it always protected by at least one key K whose corresponding decryption key K^{-1} is not penetrable?

The first two questions must be answered affirmatively to apply Authentication Tests 1–3, which then entail that there exist matching regular transforming edges.

But must those regular transforming edges lie on the strands that we expect them to (Question 3)? A common cause of authentication failure arises when there is also another edge that can transform the same value (e.g. Neuman-Stubblebine with re-authentication and Woo-Lam). Alternatively, we may know that a transforming edge of the kind desired is present, but it may not determine all of the parameters that we would like to agree on (Question 4). This was the reason for the failure of the original Needham-Schroeder protocol, and for the second Woo-Lam failure.

If the third and fourth questions are answered affirmatively, then the authentication goals of the protocol will have been met. Finally, question 5 assures that the protocol's secrecy goals will also be met.

Protocol designers need to be alert when Question 3 and Question 4 receive negative answers. Then there are *unintended services*, situations in which the protocol itself offers a transformation that can be abused by the penetrator. We recommend that protocol designers, even when working without any formal framework, ask themselves whether their protocols offer any unintended services to assist the penetrator in achieving what the protocol regards as establishing authentication. Unintended services are easy to recognize, and they are a strong clue where an attack on a protocol may lie.

Acknowledgments We are grateful to Sylvan Pinsky and Al Maneki for encouragement, support, and many technical discussions. We are grateful to Jonathan Herzog for suggesting that we develop these ideas from the germinal form they had in another paper. He and Lenore Zuck also helped us to improve the content of the paper.

References

[1] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *Proceedings of the Royal Society*, Series A, 426(1871):233–271, December 1989. Also appeared as SRC Research Report 39 and, in a shortened form, in ACM Transactions on Computer Systems 8, 1 (February 1990), 18-36.

[2] I. Cervesato, N. A. Durgin, P. D. Lincoln, J. C. Mitchell, and A. Scedrov. A meta-notation for protocol analysis. In *Proceedings, 12th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, June 1999.

[3] J. Clark and J. Jacob. A survey of authentication protocol literature: Version 1.0. University of York, Department of Computer Science, November 1997.

[4] E. Clarke, S. Jha, and W. Marrero. Using state space exploration and a natural deduction style message derivation engine to verify security protocols. In *Proceedings, IFIP Working Conference on Programming Concepts and Methods (*PROCOMET*)*, 1998.

[5] T. Dierks and C. Allen. The TLS protocol. RFC 2246, January 1999.

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

[7] A. Durante, R. Focardi, and R. Gorrieri. CVS: A compiler for the analysis of cryptographic protocols. In *12th Computer Security Foundations Workshop Proceedings*, pages 203–212. IEEE Computer Society Press, June 1999.

[8] J. D. Guttman and F. J. THAYER Fábrega. Authentication tests and the normal penetrator. MTR 00B04, The MITRE Corporation, February 2000. Also submitted for publication.

[9] T. Hwang, N.-Y. Lee, C.-M. Li, M.-Y. Ko, and Y.-H. Chen. Two attacks on Neuman-Stubblebine authentication protocols. *Information Processing Letters*, 53:103–107, 1995.

[10] G. Lowe. An attack on the Needham-Schroeder public key authentication protocol. *Information Processing Letters*, 56(3):131–136, Nov. 1995.

[11] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Proceeedings of* TACAS, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166. Springer Verlag, 1996.

[12] G. Lowe. Casper: A compiler for the analysis of security protocols. In *10th Computer Security Foundations Workshop Proceedings*, pages 18–30. IEEE Computer Society Press, 1997.

[13] W. Marrero, E. Clarke, and S. Jha. A model checker for authentication protocols. In C. Meadows and H. Orman, editors, *Proceedings of the DIMACS Workshop on Design and Verification of Security Protocols*. DIMACS, Rutgers University, September 1997.

[14] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12), Dec. 1978.

[15] B. C. Neuman and S. G. Stubblebine. A note on the use of timestamps as nonces. *Operating Systems Review*, 27(2):10–14, Apr. 1993.

[16] D. Otway and O. Rees. Efficient and timely mutual authentication. *Operating Systems Review*, 21(1):8–10, Jan. 1987.

[17] L. C. Paulson. Proving properties of security protocols by induction. In *10th IEEE Computer Security Foundations Workshop*, pages 70–83. IEEE Computer Society Press, 1997.

[18] D. X. Song. Athena: a new efficient automated checker for security protocol analysis. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, June 1999.

[19] M. Tatebayashi, N. Matsuzaki, and D. Newman. Key distribution protocol for digital mobile communication systems. In G. Brassard, editor, *Advances in Cryptology:* CRYPTO *'89*, volume 435 of *Lecture Notes in Computer Science*, pages 324–331. Springer Verlag, 1990.

[20] F. J. THAYER Fábrega, J. C. Herzog, and J. D. Guttman. Mixed strand spaces. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, June 1999.

[21] F. J. THAYER Fábrega, J. C. Herzog, and J. D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(2/3):191–230, 1999.

[22] T. Y. C. Woo and S. S. Lam. Authentication for distributed systems. *Computer*, 25(1):39–52, Jan. 1992.

[23] T. Y. C. Woo and S. S. Lam. A lesson on authentication protocol design. *Operating Systems Review*, pages 24–37, 1994.

A Strands, Bundles, and the Penetrator

In this appendix, we will introduce the basic strand space notions to be used in the remainder of the paper. This material is derived from [21, 20], with a few small changes. For instance, the principle of induction on terms was previously inadvertently omitted from the freeness axiom (Axiom 1). The penetrator strands of type T were unnecessary and have now been eliminated from Definition A.9.

A.1 Strand Spaces

Consider a set A, the elements of which are the possible messages that can be exchanged between principals in a protocol. We will refer to the elements of A as *terms*. We assume that a *subterm* relation is defined on A. $t_0 \sqsubset t_1$ means t_0 is a subterm of t_1. We constrain the set A further below in Section A.3, and define a subterm relation there.

In a protocol, principals can either send or receive terms. We represent transmission of a term as the occurrence of that term with positive sign, and reception of a term as its occurrence with negative sign.

Definition A.1 *A* signed term *is a pair* $\langle \sigma, a \rangle$ *with* $a \in \mathsf{A}$ *and* σ *one of the symbols* $+, -$. *We will write a signed term as* $+t$ *or* $-t$. $(\pm\mathsf{A})^*$ *is the set of finite sequences of signed terms. We will denote a typical element of* $(\pm\mathsf{A})^*$ *by* $\langle \langle \sigma_1, a_1 \rangle, \ldots, \langle \sigma_n, a_n \rangle \rangle$.

A strand space *over* A *is a set* Σ *together with a trace mapping* $\mathrm{tr} : \Sigma \to (\pm\mathsf{A})^*$.

By abuse of language, we will still treat signed terms as ordinary terms. For instance, we shall refer to subterms of signed terms. We will usually represent a strand space by its underlying set of strands Σ.

Definition A.2 Fix a strand space Σ.

1. A *node* is a pair $\langle s, i \rangle$, with $s \in \Sigma$ and i an integer satisying $1 \leq i \leq \mathrm{length}(\mathrm{tr}(s))$. The set of nodes is denoted by $\mathcal{N}$. We will say the node $\langle s, i \rangle$ belongs to the strand s. Clearly, every node belongs to a unique strand.

2. If $n = \langle s, i \rangle \in \mathcal{N}$ then $\mathrm{index}(n) = i$ and $\mathrm{strand}(n) = s$. Define $\mathrm{term}(n)$ to be $(\mathrm{tr}(s))_i$, i.e. the ith signed term in the trace of s. Similarly, $\mathrm{uns\_term}(n)$ is $((\mathrm{tr}(s))_i)_2$, i.e. the unsigned part of the ith signed term in the trace of s.

3. There is an edge $n_1 \rightarrow n_2$ if and only if $\mathrm{term}(n_1) = +a$ and $\mathrm{term}(n_2) = -a$ for some $a \in \mathsf{A}$. Intuitively, the edge means that node n_1 sends the message a, which is received by n_2, recording a potential causal link between those strands.

4. When $n_1 = \langle s, i \rangle$ and $n_2 = \langle s, i+1 \rangle$ are members of $\mathcal{N}$, there is an edge $n_1 \Rightarrow n_2$. Intuitively, the edge expresses that n_1 is an immediate causal predecessor of n_2 on the strand s. We write $n' \Rightarrow^+ n$ to mean that n' precedes n (not necessarily immediately) on the same strand.

5. An unsigned term t *occurs in* $n \in \mathcal{N}$ iff $t \sqsubset \mathrm{term}(n)$.

6. Suppose I is a set of unsigned terms. The node $n \in \mathcal{N}$ is an *entry point* for I iff $\mathrm{term}(n) = +t$ for some $t \in I$, and whenever $n' \Rightarrow^+ n$, $\mathrm{term}(n') \notin I$.

7. An unsigned term t *originates* on $n \in \mathcal{N}$ iff n is an entry point for the set $I = \{t' : t \sqsubset t'\}$.

8. An unsigned term t is *uniquely originating* iff t originates on a unique $n \in \mathcal{N}$.

If a term t originates uniquely in a particular strand space, then it can play the role of a nonce or session key in that structure.

$\mathcal{N}$ together with both sets of edges $n_1 \rightarrow n_2$ and $n_1 \Rightarrow n_2$ is a directed graph $\langle \mathcal{N}, (\rightarrow \cup \Rightarrow) \rangle$.

A.2 Bundles and Causal Precedence

A *bundle* is a finite subgraph of $\langle \mathcal{N}, (\rightarrow \cup \Rightarrow) \rangle$, for which we can regard the edges as expressing the causal dependencies of the nodes.

Definition A.3 *Suppose $\rightarrow_{\mathcal{C}} \subset \rightarrow$; suppose $\Rightarrow_{\mathcal{C}} \subset \Rightarrow$; and suppose $\mathcal{C} = \langle \mathcal{N}_{\mathcal{C}}, (\rightarrow_{\mathcal{C}} \cup \Rightarrow_{\mathcal{C}}) \rangle$ is a subgraph of $\langle \mathcal{N}, (\rightarrow \cup \Rightarrow) \rangle$. $\mathcal{C}$ is a bundle if:*

1. *$\mathcal{N}_{\mathcal{C}}$ and $\rightarrow_{\mathcal{C}} \cup \Rightarrow_{\mathcal{C}}$ are finite.*

2. *If $n_2 \in \mathcal{N}_{\mathcal{C}}$ and $\mathrm{term}(n_2)$ is negative, then there is a unique n_1 such that $n_1 \rightarrow_{\mathcal{C}} n_2$.*

3. *If $n_2 \in \mathcal{N}_{\mathcal{C}}$ and $n_1 \Rightarrow n_2$ then $n_1 \Rightarrow_{\mathcal{C}} n_2$.*

4. *$\mathcal{C}$ is acyclic.*

In conditions 2 and 3, it follows that $n_1 \in \mathcal{N}_{\mathcal{C}}$, because $\mathcal{C}$ is a graph.

For our purposes, it does not matter whether communication is regarded as a synchronizing event or as an asynchronous activity. The definition of bundle formalizes a process communication model with three properties:

- A strand (process) may send and receive messages, but not both at the same time;
- When a strand receives a message t, there is a unique node transmitting t from which the message was immediately received;
- When a strand transmits a message t, many strands may immediately receive t.

Notational Convention A.4 *A node n is in a bundle $\mathcal{C} = \langle \mathcal{N}_{\mathcal{C}}, \rightarrow_{\mathcal{C}} \cup \Rightarrow_{\mathcal{C}} \rangle$, written $n \in \mathcal{C}$, if $n \in \mathcal{N}_{\mathcal{C}}$; a strand s is in $\mathcal{C}$ if all of its nodes are in $\mathcal{N}_{\mathcal{C}}$.*

If $\mathcal{C}$ is a bundle, then the $\mathcal{C}$-height *of a strand s is the largest i such that $\langle s, i \rangle \in \mathcal{C}$.* $\mathcal{C}$-trace$(s) = \langle tr(s)(1), \ldots, tr(s)(m) \rangle$, *where $m = \mathcal{C}$-height(s).*

Definition A.5 *If $\mathcal{S}$ is a set of edges, i.e. $\mathcal{S} \subset \rightarrow \cup \Rightarrow$, then $\prec_{\mathcal{S}}$ is the transitive closure of $\mathcal{S}$, and $\preceq_{\mathcal{S}}$ is the reflexive, transitive closure of $\mathcal{S}$.*

The relations $\prec_{\mathcal{S}}$ and $\preceq_{\mathcal{S}}$ are each subsets of $\mathcal{N}_{\mathcal{S}} \times \mathcal{N}_{\mathcal{S}}$, where $\mathcal{N}_{\mathcal{S}}$ is the set of nodes incident with any edge in $\mathcal{S}$.

Proposition A.6 *Suppose $\mathcal{C}$ is a bundle. Then $\preceq_{\mathcal{C}}$ is a partial order, i.e. a reflexive, antisymmetric, transitive relation. Every non-empty subset of the nodes in $\mathcal{C}$ has $\preceq_{\mathcal{C}}$-minimal members.*

We regard $\preceq_{\mathcal{C}}$ as expressing causal precedence, because $n \prec_{\mathcal{S}} n'$ holds only when n's occurrence causally contributes to the occurrence of n'. When a bundle $\mathcal{C}$ is understood, we will simply write $\preceq$. Similarly, "minimal" will mean $\preceq_{\mathcal{C}}$-minimal.

A.3 Terms, Encryption, and Freeness Assumptions

We will now specialize the set of terms A. In particular we will assume given:

- A set $\mathsf{T} \subseteq \mathsf{A}$ of texts (representing the atomic messages).
- A set $\mathsf{K} \subseteq \mathsf{A}$ of cryptographic keys disjoint from T, equipped with a unary operator **inv** : $\mathsf{K} \rightarrow \mathsf{K}$. We assume that **inv** is an inverse mapping each member of a key pair for an asymmetric cryptosystem to the other, and each symmetric key to itself.
- Two binary operators **encr** : $\mathsf{K} \times \mathsf{A} \rightarrow \mathsf{A}$ and **join** : $\mathsf{A} \times \mathsf{A} \rightarrow \mathsf{A}$.

We will follow custom and write $\mathbf{inv}(K)$ as K^{-1}, $\mathbf{encr}(K, m)$ as $\{|m|\}_K$, and $\mathbf{join}(a, b)$ as $a\, b$. If $\mathfrak{K}$ is a set of keys, $\mathfrak{K}^{-1}$ denotes the set of inverses of elements of $\mathfrak{K}$. We assume, like many others (e.g. [12, 13, 17]), that A is freely generated, which is crucial for the results in this paper.

Axiom 1 A *is freely generated from* T *and* K *by* **encr** *and* **join**.

Definition A.7 *The subterm relation $\sqsubset$ is defined inductively, as the smallest relation such that $a \sqsubset a$; $a \sqsubset \{|g|\}_K$ if $a \sqsubset g$; and $a \sqsubset g\, h$ if $a \sqsubset g$ or $a \sqsubset h$.*

By this definition, for $K \in \mathsf{K}$, we have $K \sqsubset \{|g|\}_K$ only if $K \sqsubset g$ already.

Definition A.8 *1. If $\mathfrak{K} \subset \mathsf{K}$, then $t_0 \sqsubset_{\mathfrak{K}} t$ if t is in the smallest set containing t_0 and closed under encryption with $K \in \mathfrak{K}$ and concatenation with arbitrary terms t_1.*

2. *A term t is* simple *if it is not of the form $g\,h$.*

3. *A term t_0 is a* component *of t if t_0 is simple and $t_0 \sqsubset_\emptyset t$.*

A.4 Penetrator Strands

The atomic actions available to the penetrator are encoded in a set of *penetrator traces*. They summarize his ability to discard messages, generate well known messages, piece messages together, and apply cryptographic operations using keys that become available to him. A protocol attack typically requires hooking together several of these atomic actions.

The actions available to the penetrator are relative to the set of keys that the penetrator knows initially. We encode this in a parameter, the set of penetrator keys $\mathsf{K}_\mathcal{P}$.

Definition A.9 *A* penetrator trace *relative to $\mathsf{K}_\mathcal{P}$ is one of the following:*

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

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

$\mathbf{C}_{g,h}$ *Concatenation:* $\langle -g,\ -h,\ +g\,h \rangle$

$\mathbf{S}_{g,h}$ *Separation:* $\langle -g\,h,\ +g,\ +h \rangle$

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

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

$\mathcal{P}_\Sigma$ is the set of all strands $s \in \Sigma$ such that $\mathrm{tr}(s)$ is a penetrator trace.

A strand $s \in \Sigma$ is a penetrator strand *if it belongs to $\mathcal{P}_\Sigma$, and a node is a* penetrator node *if the strand it lies on is a penetrator strand. Otherwise we will call it a* non-penetrator *or* regular *strand or node. A node n is* **M**, **C**, *etc. node if n lies on a penetrator strand with a trace of kind* **M**, **C**, *etc.*

Contents