Electronic Notes in Theoretical Computer Science 47 (2001)
URL: http://www.elsevier.nl/locate/entcs/volume47.html pp. 1 – 21

Key Compromise, Strand Spaces, and the Authentication Tests ⋆

Joshua D. Guttman

The MITRE Corporation
Bedford, MA 01730 USA
guttman@mitre.org

Abstract

Some cryptographic protocols are vulnerable to replay attacks, a type of weakness that was a focus of attention in the Burroughs-Abadi-Needham logic. Newer, more operational approaches to protocol analysis have not concentrated on this type of attack. This paper fills the gap for the strand space theory.

The main technical point is to provide a definition of recency. Our candidate is convenient because we already have a powerful way to prove events recent, namely the incoming and outgoing authentication tests.

A secondary purpose of this paper is to illustrate an easily mechanized pattern for using the authentication tests.

Key words: cryptographic protocols, authentication, authentication tests, strand spaces, bundles, key compromise

1 Introduction

Some cryptographic protocols are vulnerable to replay attacks, in which a message from a legitimate protocol execution is retransmitted later, typically when the same principal executes a different session of the protocol. If the message contains an encrypted session key that a penetrator has cracked via cryptanalysis, then the penetrator may induce the recipient to reuse the compromised session key.

Although the Burroughs-Abadi-Needham logic [1] was particularly successful at identifying these situations, the newer, more operational approaches to protocol analysis have not concentrated on this type of attack. These include strand spaces [7,19], CSP model-checking [9], and rank functions [8,18], while Paulson's inductive method [15,16] was the primary exception to this.

⋆ I gave my MFPS 17 talk on 24 May 2001, Bob Dylan's sixtieth birthday. This paper is dedicated to him.

In this paper we fill this gap in the strand space literature. We show how to specify the behavior of a key server so that recent keys remain uncompromised, whereas older keys may have been compromised. The familiar Needham-Schroeder private key protocol [13] is vulnerable in this model, whereas the Yahalom protocol [1] is not. The main technical point is to provide a definition of recency. Our candidate is convenient because the strand space method already has a powerful way to prove that an event is recent, namely the incoming and outgoing authentication tests [5,7].

A supplementary purpose of this paper is to illustrate the power of the authentication tests. They provide simple and straightforward explanations of protocol correctness. We linger over several examples to illustrate the patterns of usage for the authentication tests. These patterns of usage are quite stereotyped, although there are several ingredients to be combined effectively.

The Needham-Schroeder Symmetric Key Protocol In the seminal paper [13], Needham and Schroeder proposed a protocol using symmetric cryptography and another protocol using public key cryptography. Both turned out to be flawed.

The symmetric-key protocol NSSK assumes that each principal shares a long term key with a key server. Its purpose is to establish a secret session key between two principals, after each has authenticated the identity of the other. The intended execution of the protocol is summarized in Figure 1.

In this protocol, the initiator A sends a message to the key server containing his name, the name B of his partner, and a nonce N_a. A nonce is a new, random number, generated for just one use. The message is encrypted using a long-term key, shared between A and the server. The server generates a new session key K, and sends it encrypted with A's long-term key. B's name and the nonce N_a are enclosed with the session, together with a ticket. The ticket contains A's name and the session key, encrypted with B's long-term key. A forwards this ticket to B, expecting to receive back a new nonce N_b, encrypted with K. A demonstrates receipt by returning $1 + N_b$, also encrypted with K.

The shape of the diagram in Figure 1 is striking: It consists of two rectangles, connected by a "neck," the middle $\Rightarrow$ in A's behavior. The top rectangle involves just A and the server, while the bottom rectangle involves just A and B, so that the server is completed before B begins.

Denning and Sacco [2] promptly identified the failure that results. Although B, when receiving $\{\!|A\ ,\ K|\!\}_{K_B}$, can infer that it was generated by the server, B has no way to know how much time has elapsed since the server completed. This is geometrically evident from Figure 1, as nothing available to B measures the length of the "neck" between A's second and third nodes, nor the delay along the $\rightarrow$ arrow from A's third node to B's first node. So B cannot bound the delay between when S generated the session key K and when B receives it.

Assuming that A and B use K in a session for encrypted communication,

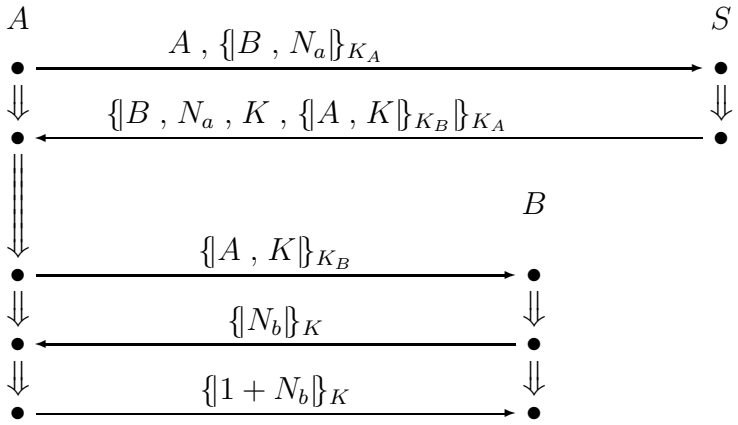


Fig. 1. The Needham-Schroeder Symmetric Key Protocol (NSSK)

perhaps a penetrator will cryptanalyze the conversation to recover K. In this case, the penetrator can start a session by resending the same unit $\{|A\ ,\ K|\}_{K_B}$, later returning $\{|1+N_b|\}_K$. Then B will have been had. The assumption that K is good should expire when it is no longer recently generated.

Our goal in this paper is to formulate a rigorous theory allowing us to prove that a protocol does not have flaws of this kind. The same ideas also suggest heuristics for finding such flaws, when we cannot prove their absence.

2 Strand Spaces

The strand space theory [7,19] is a particular way to formalize the Dolev-Yao model [3]. It assumes that cryptography is ideal, and aims to discover what authentication and secrecy properties protocols then achieve.

In the strand space model, the behaviors of principals are represented by a set of *strands*. A strand is a sequence of message send and receive events; for instance, each vertical column in Figure 1 is a strand (for each particular choice of parameters A, B, N_a, etc.). The behaviors of the penetrator are also represented by strands; in this case, the strands represent basic actions such as encrypting or decrypting with a known key or selecting a nonce. Each strand is a purely local behavior of some particular principal on a particular occasion. A collection of strands forms a *bundle* if they are causally well-founded, so that every message received was actually sent previously by some strand, and no principal engages in a second or third event without having already performed the first.

2.1 Background

We compactly summarize the ideas behind the strand space model [4,7,19]; definitions are contained in Appendix A.

Terms and Subterms A is the set of messages that can be sent between principals. Its elements, called *terms*, are freely generated from two disjoint sets, T (texts such as nonces or names) and K (keys), by 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.)

In NSSK, the initiator A sends a term of the form $A\ ,\ \{\!|B\ ,\ N_a|\!\}_{K_A}$ to start an exchange intended for B. This is a concatenation containing A's name and a ciphertext. The ciphertext is created using A's long-term key K_A shared with the key server; its plaintext is the result of concatenating B's name with a nonce (random bitstring) N_a.

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.) For instance, the subterms of $A\ ,\ \{\!|B\ ,\ N_a|\!\}_{K_A}$ are A, B, N_a, the concatenated message $B\ ,\ N_a$, the ciphertext $\{\!|B\ ,\ N_a|\!\}_{K_A}$, and the whole term itself. The key K_A is not part of what is uttered; it just contributes to *how* the message is constructed.

Strands and Strand Spaces 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, $s \downarrow i$ is the i^{th} node on s. The relation $n \Rightarrow n'$ holds between nodes n and n' if $n = s \downarrow i$ and $n' = s \downarrow i+1$. Hence, $n \Rightarrow^+ n'$ means that $n = s \downarrow i$ and $n' = s \downarrow j$ for some $j > i$. The relation $n \rightarrow n'$ represents inter-strand communication; it means that $\mathrm{term}(n) = +t$ and node $\mathrm{term}(n') = -t$.

Continuing with NSSK as our illustration, an initiator strand offers a sequence of events of the form shown in the leftmost column of Figure 1. In this strand s_i, the initiator A sends a term intended for the server S, and expects to receive back a "ticket" of the form $\{\!|B\ ,\ N_a\ ,\ K\ ,\ \{\!|A\ ,\ K|\!\}_{K_B}|\!\}_{K_A}$, after which it will forward $\{\!|A\ ,\ K|\!\}_{K_B}$, and so on. The first reception is $s_i \downarrow 2$ and the next transmission is $s_i \downarrow 3$.

An NSSK initiator strand has five parameters (or degrees of freedom), namely the two names A and B, the two nonces N_a and N_b, and the session key K. When we write $s_i \in \mathrm{Init}[A, B, N_a, N_b, K]$ in this illustration, we will mean that s_i is an initiator strand using the particular values shown as parameters, and similarly for $s_s \in \mathrm{Serv}[A, B, N_a, K]$ and $s_r \in \mathrm{Resp}[A, B, N_b, K]$. A is the principal active in $\mathrm{Init}[A, B, N_a, N_b, K]$ as initiator, while B is the principal active in $\mathrm{Resp}[A, B, N_b, K]$ as responder. We generally write parameters in this order: first the names of the intended participants, next any nonces seen in that run, and finally a session key (if the protocol distributes one).

A strand represents the local view of a participant. For a legitimate participant, a strand 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.

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 $\rightarrow$.

Origination We say that a term t *originates* at a node $n = s \downarrow i$ if the sign of n is positive; $t \sqsubset \text{term}(n)$; and $t \not\sqsubset \text{term}(s \downarrow i')$ for every $i' < i$. Thus, n represents a message transmission that includes t, and it is the first node in s including t. For instance, if $s_i \in \text{Init}[A, B, N_a, N_b, K]$, then N_a and A both originate at $s_i \downarrow 1$.

If a value originates on only one node in a set S of nodes, we call it *uniquely originating* in S; uniquely originating values are desirable as nonces and session keys. When we assume that a value like N_a originates uniquely in some set S, we are effectively assuming that S is not unrealistically large, in a particular sense, namely so large as to contain independent events in which the same value is repeatedly chosen at random from a large set. Although both N_a and A originate at $s_i \downarrow 1$, N_a is more likely to originate *uniquely* in relevant sets of nodes S.

Bundles A *bundle* is a finite, causally well-founded graph of nodes and arrows of both kinds $\rightarrow$ and $\Rightarrow$. 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. If $s \downarrow i$ is in a bundle and $j < i$, then $s \downarrow j$ must be in it also; taking the i^{th} step depends on having already taken all previous steps. Moreover, the arrows must be acyclic, since otherwise an event would have preceded itself. (See Definition A.3.)

By acyclicity, the relation $n_0 \preceq n_1$ between nodes, which holds when there is a sequence of edges leading from n_0 to n_1, is a partial ordering. Since bundles are finite, $\preceq$ is in fact a well founded relation, which justifies a principle of bundle induction (Proposition A.6). Every execution of a protocol must consist of a bundle, because a run is possible only if the causal constraints formalized in the definition of bundle are satisfied.

The height of a strand in a bundle is the number of nodes on the strand that are in the bundle. Authentication theorems assert that if some regular strand has at least a given height in a bundle, meaning that the principal must have engaged in at least that many steps of its run, then another regular strand must have a certain height in the bundle. They often assume that certain values are uniquely originating in the nodes of that bundle.

The Penetrator While regular principals are represented only by what they say and hear, the behavior of the penetrator is represented more explicitly. The values he deduces are treated as if they had been said publicly. A penetrator strand represents an atomic deduction, and more complex actions use several penetrator strands. 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; M-strands emit known atomic texts or guesses; and K-strands emit keys from a set of known keys. The set of initially known keys, that may be emitted in K-strands, is called $\mathsf{K}_{\mathcal{P}}$. (See Definition A.8.)

2.2 Key Secrecy

In most cryptographic protocols, key secrecy relies on two basic facts. We let $\mathcal{B}$ be a bundle, and consider how the regular nodes in $\mathcal{B}$ transmit keys.

First, suppose no regular strand in $\mathcal{B}$ ever originates a term containing a key K—that is, transmits K without having received it previously in the same form. Then either the penetrator starts off knowing K in $\mathsf{K}_{\mathcal{P}}$, or else the penetrator never learns it. This is typically the situation with long-term shared keys, or with private keys in asymmetric cryptography. If $K \notin \mathsf{K}_{\mathcal{P}}$ and no regular node in $\mathcal{B}$ ever originates a term containing K, then we say that K is *immediately safe* in $\mathcal{B}$, written $K \in \mathsf{S}_0(\mathcal{B})$, or $K \in \mathsf{S}_0$, since we will normally suppress the dependency of S_0 on $\mathcal{B}$.

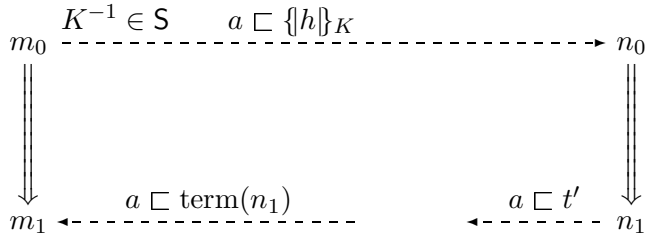
Second, suppose that a key K may be transmitted by a regular strand in $\mathcal{B}$, but if so it is always encrypted using another key K' such that $K' \in \mathsf{S}_0$. Then either the penetrator starts off knowing K in $\mathsf{K}_{\mathcal{P}}$, or else the penetrator never learns it. This is typically the situation with session keys in well-designed protocols. When $K \notin \mathsf{K}_{\mathcal{P}}$ and every regular strand that ever originates K protects it with some $K' \in \mathsf{S}_0$, then we say that K is *safe at level 1*, written $K \in \mathsf{S}_1(\mathcal{B})$. Again, typically no confusion results from omitting $\mathcal{B}$.

An inductive definition would be possible, but in practice S_0 and S_1 are usually sufficient. The *safe keys* are $\mathsf{S} = \mathsf{S}_0 \cup \mathsf{S}_1$. In [7] we prove that if $K \in \mathsf{S}$ and $\mathcal{B}$ is a bundle, then there is no node $n \in \mathcal{B}$ such that $K = \mathrm{term}(n)$.

2.3 The Authentication Tests

Suppose in some bundle $\mathcal{B}$, a principal creates and transmits a uniquely originating value a, such as a nonce, and later receives a back in some cryptographically altered form. Then we could follow the trajectory of the value a from its origin, which we can call m_0, through various sends and receives and along various strands, until it arrives back at the node m_1 later on the same strand. Since m_1 contains a in cryptographically altered form, somewhere along this trajectory, someone has transformed it. If the key involved is safe, that someone cannot be the penetrator, it can only be a regular principal. This is the *authentication test* idea, as developed and justified in [4,7,17]. There are two main forms of authentication test.

Outgoing Tests A uniquely originating value a may be transmitted only in encrypted form $\{\!|\ldots a\ldots|\!\}_K$ where the decryption key $K^{-1} \in \mathsf{S}$ is safe. If it



Assume $\{\!|h|\!\}_K \not\sqsubset \mathrm{term}(n_1)$

a originates uniquely at m_0, contained only in $\{\!|h|\!\}_K$

Conclude nodes n_0, n_1 exist in $\mathcal{B}$ and are regular

$\{\!|h|\!\}_K \not\sqsubset t'$

$m_0 \prec n_0 \prec n_1 \prec m_1$

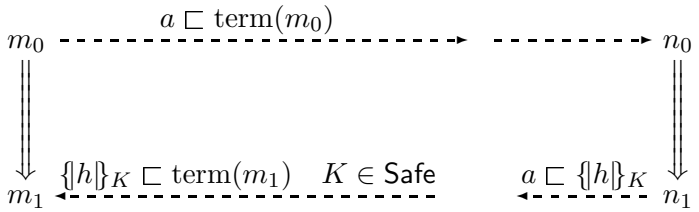
Fig. 2. Outgoing Authentication Test

is later received outside the context $\{\!|\ldots a \ldots|\!\}_K$, then a regular participant, not the penetrator, must have been responsible the first time it appears in a different context. Observe that this regular participant transforms a after the original transmission of $\{\!|\ldots a \ldots|\!\}_K$ at m_0 and before the transformed version is received back at m_1. The temporal relations "after" and "before" refer to the ordering $\prec_{\mathcal{B}}$ generated by the arrows in the bundle $\mathcal{B}$ (as in Definition A.5). It is an *outgoing test* because the encrypted unit goes out; see Figure 2.

Incoming Tests If, instead, a is received in encrypted form $\{\!|\ldots a \ldots|\!\}_K$ although it was not sent in that context, and the encryption key $K \in \mathsf{S}$ is safe, then a regular participant must have been responsible when a entered this context. Again, the transformation producing $\{\!|\ldots a \ldots|\!\}_K$ must occur after m_0 and before m_1. We call this an *incoming test* because the encrypted unit comes in, as shown in Figure 3.

Sometimes a uniquely originating value a is transmitted in one encrypted form $\{\!|h|\!\}_K$ and received back in a different $\{\!|h'|\!\}_{K'}$. We include cases where $h = h'$ or $K = K'$, though not both. If $K^{-1} \in \mathsf{S}$ and $K' \in \mathsf{S}$, then this is *both* an outgoing test and an incoming test. However, these two views may have different consequences. As an outgoing test, it implies a regular transforming edge that accepts $\{\!|h|\!\}_K$ and extracts a from it. This may be of some form other than $\{\!|h'|\!\}_{K'}$, since another principal may later transform it again. The incoming test yields a transforming edge creating $\{\!|h'|\!\}_{K'}$, although it may have received a in a form other than $\{\!|h|\!\}_K$.

Pattern Matching to Find Transforming Edges The authentication tests are important because they show that a *regular* strand does the transformation. Typically, there are very few edges on regular strands that are syntactically capable of performing the transformation. So a simple pattern



Assume $\{\!|h|\!\}_K \not\sqsubset \mathrm{term}(m_0)$

a originates uniquely at m_0

Conclude nodes n_0, n_1 exist in $\mathcal{B}$ and are regular

$m_0 \prec n_0 \prec n_1 \prec m_1$

Fig. 3. Incoming Authentication Test

matching determines which regular strands could have been responsible.

We call the responsible strand a *transforming edge*, because it transforms the way that a occurs. We will use the symbol $\rightsquigarrow$ to summarize what the transforming edge must do. We write for instance $\{\!|B\ ,\ N_a|\!\}_{K_A} \rightsquigarrow N_a$ for a regular transforming edge containing a negative node with subterm $\{\!|B\ ,\ N_a|\!\}_{K_A}$ followed by a positive node emitting a message in which N_a occurs *outside* the context $\{\!|B\ ,\ N_a|\!\}_{K_A}$. An outgoing test guarantees the existence of a regular, non-penetrator edge $\{\!|B\ ,\ N_a|\!\}_{K_A} \rightsquigarrow N_a$. The $\rightsquigarrow$ notation does not make explicit the form in which N_a occurs; however, it is required that it occurs outside the context shown in the explicitly encrypted term on the other side.

Conversely, we write $N_a \rightsquigarrow \{\!|B\ ,\ N_a|\!\}_{K_A}$ for a regular transforming edge containing a negative node receiving N_a outside of $\{\!|B\ ,\ N_a|\!\}_{K_A}$, followed by a positive node emitting a message with subterm $\{\!|B\ ,\ N_a|\!\}_{K_A}$. An incoming test establishes that a regular transforming edge of this form exists.

Outgoing Tests: Additional Conclusion There are additional conclusions that an authentication test allows in certain cases (see [7, Section 4.2.1]); one is relevant to the Yahalom protocol considered below.

Suppose in Figure 2 that $a \sqsubset \{\!|h_1|\!\}_{K_1} \sqsubset \mathrm{term}(n_1)$, that a occurs only within $\{\!|h_1|\!\}_{K_1}$ in $\mathrm{term}(n_1)$, and that $K_1^{-1} \in \mathsf{S}$. Then there is also a *regular* node n_2 receiving $\{\!|h_1|\!\}_{K_1}$, meaning n_2 is negative and $\{\!|h_1|\!\}_{K_1} \sqsubset \mathrm{term}(n_2)$.

Why does this hold? Either $\{\!|h_1|\!\}_{K_1} \sqsubset \mathrm{term}(m_1)$, in which case m_1 is such a node, or else some principal must transform $\{\!|h_1|\!\}_{K_1}$ to put a into whatever form it appears in $\mathrm{term}(m_1)$. Because $K_1^{-1} \in \mathsf{S}$, this transformation can be done only by a regular strand, which must receive $\{\!|h_1|\!\}_{K_1}$ to do so.

This principle often helps to constrain the term $\{\!|h_1|\!\}_{K_1}$, or to fill in some parameters of the receiving strand. We use this principle twice in Section 5.3.

Unsolicited Tests A third, related but weaker, type of test is the *unso-*

Test	Test edge	Constraint	Transforming edge	Bound
Outgoing	$+\{\|h\|\}_K \Rightarrow -\ldots a\ldots$	$K^{-1} \in \mathsf{S}$	$\{\|h\|\}_K \rightsquigarrow a$	$\times$
Incoming	$+\ldots a\ldots \Rightarrow -\{\|h\|\}_K$	$K \in \mathsf{S}$	$a \rightsquigarrow \{\|h\|\}_K$	$\times$
Unsolicited	$-\{\|h\|\}_K$	$K \in \mathsf{S}$	$\rightsquigarrow \{\|h\|\}_K$	

Table 1
The Authentication Tests

licited test. If a term $\{\|t\|\}_K$ is received, and $K \in \mathsf{S}$ is safe, then $\{\|t\|\}_K$ originated on some regular strand. After all, it originated somewhere, and that can not have been a penetrator strand if $K \in \mathsf{S}$. Here we know only that the regular node originating $\{\|t\|\}_K$ is before the node on which it is received. We do not know any node after which it must have occurred. We write $\rightsquigarrow \{\|B\ ,\ N_a\|\}_{K_A}$ for the positive node that must exist as a result of an unsolicited test.

The authentication tests are summarized in Table 1. The last column contains $\times$ if the first node on the test edge is a lower bound (in the ordering $\preceq$) constraining when the transforming edge occurs.

2.4 Assumptions for This Paper

If either of the long-term keys of two principals A and B is compromised, meaning that either $K_A \in \mathsf{K}_\mathcal{P}$ or $K_B \in \mathsf{K}_\mathcal{P}$, then symmetric-key protocol exchanges involving them are hopeless: No security guarantee of interest will hold. Thus, we assume henceforth that for two particular principals A, B,

Assumption 1 $K_A, K_B \notin \mathsf{K}_\mathcal{P}$.

Hence, for reasonable protocols, K_A, K_B will be safe, i.e. $K_A, K_B \in \mathsf{S}_0$.

In the examples, we consider only symmetric cryptography, so we assume that the same key is used to create a ciphertext $\{\|t\|\}_K$ as to decrypt it:

Assumption 2 *For all keys K, $K = K^{-1}$.*

3 Key Servers

We now formalize the intended behavior of key servers. They must obey three principles. First, the session key K delivered must never be known initially to the penetrator, since otherwise K is certainly not safe. Second, the key server must not re-use a session key, since it might spoil the confidentiality of a session key by retransmitting the same key to a compromised principal. A third principle is that a session key is never the same as a principal's long term key, since otherwise the server might disclose a long term key by sending it (as a session key) to a compromised principal. We summarize these principles:

KS1 If $\exists s\ .\ s \in \mathrm{Serv}[**, K]$ then $K \notin \mathsf{K}_\mathcal{P}$;

KS2 If $s, s' \in \mathrm{Serv}[**, K]$, then $s = s'$; and

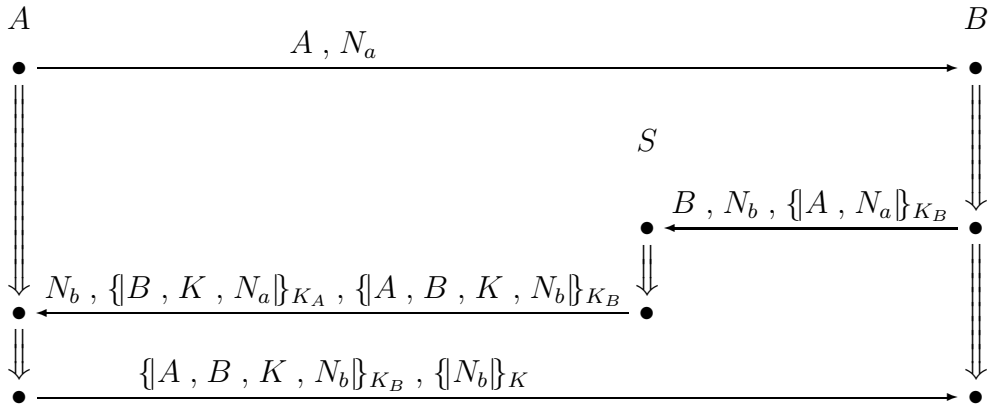


Fig. 4. The Yahalom-Paulson Protocol

KS3 $\exists s \,.\, s \in \mathrm{Serv}[**, K]$ implies $K \neq K_A$ for any A.

In normal protocols, all of the server parameters $X_1, \ldots, X_n, K$ appear non-vacuously in messages, and K originates on a node of $s \in \mathrm{Serv}[**, K]$. Then **KS1** and **KS2** imply that K is uniquely originating. Moreover, when $s \in \mathrm{Serv}[X_1, \ldots, X_n, K]$ and $s' \in \mathrm{Serv}[X'_1, \ldots, X'_n, K]$, then **KS2** implies $X_1 = X'_1, \ldots, X_n = X'_n$.

We interpret the principles as a constraint on bundles. That is, we intend to consider only bundles $\mathcal{B}$ such that **KS1**–**KS3** hold. The strands s are those strands such that for some node $n = s \downarrow i$, $n \in \mathcal{B}$. Thus, $\exists s \,.\, \phi$ means that there is some strand s with at least one node in $\mathcal{B}$ such that ϕ.

We regard a bundle $\mathcal{B}$ as formalizing what might reasonably happen over a period of time (whether brief or reasonably extended) in a network of principals executing one or more protocols. The likelihood that $\mathcal{B}$ contains a counterexample to **KS1**–**KS3** is taken to be negligible if the cryptography and key-generation algorithms are well chosen and well implemented.

4 Yahalom and its Variants

We next present three variants of the Yahalom protocol [1,16]. They suggest the main idea for analyzing key compromise. We start with a variant in which key compromise is not an issue.

4.1 The Yahalom-Paulson Protocol

Paulson [16], following ideas from the BAN paper [1], modified a protocol originally invented by Yahalom. The result, as illustrated in Figure 4, is a tight protocol not vulnerable to key compromise. In this protocol, the participants get their guarantees by a succession of incoming tests.

The Initiator's Guarantees Assume given a bundle $\mathcal{B}$ containing an ini-

tiator strand $s_i \in \mathrm{Init}[A, B, N_a, N_b, K]$, and assume that the nonces N_a, N_b are uniquely originating. What follows?

A's nonce N_a is returned in the form $\{\!|B\ ,\ K\ ,\ N_a|\!\}_{K_A}$, thereby guaranteeing an edge $N_a \rightsquigarrow \{\!|B\ ,\ K\ ,\ N_a|\!\}_{K_A}$. By case analysis, this edge occurs on a server strand s_s. By matching variables, $s_s \in \mathrm{Serv}[A, B, N_a, *, K]$.

Hence, $\{\!|A\ ,\ N_a|\!\}_{K_B} \sqsubset \mathrm{term}(s_s \downarrow 1)$. We use $\{\!|A\ ,\ N_a|\!\}_{K_B}$ as an unsolicited test term, which implies that there is a positive node $\rightsquigarrow \{\!|A\ ,\ N_a|\!\}_{K_B}$. This node can occur only on $s_r \downarrow 2$ where $s_r \in \mathrm{Resp}[A, B, N_a, **]$. We have now derived the initiator's guarantees about s_s and s_r.

The Responder's Guarantees Assume next a bundle $\mathcal{B}$ containing a responder strand $s_r \in \mathrm{Resp}[A, B, N_a, N_b, K]$, and assume that the nonces N_a, N_b are uniquely originating. B's nonce N_b is returned in both the forms $\{\!|A\ ,\ B\ ,\ K\ ,\ N_b|\!\}_{K_B}$ and $\{\!|N_b|\!\}_K$. The first is an incoming test, because $K_B \in \mathsf{S}$. This guarantees a transforming edge $N_b \rightsquigarrow \{\!|A\ ,\ B\ ,\ K\ ,\ N_b|\!\}_{K_B}$, which lies on a server strand s_s. Noting the variables occurring in $\{\!|A\ ,\ B\ ,\ K\ ,\ N_b|\!\}_{K_B}$, we infer $s_s \in \mathrm{Serv}[A, B, *, N_b, K]$.

The key server assumption **KS2** implies that K is transmitted only in the forms $\{\!|A\ ,\ B\ ,\ K\ ,\ N_b|\!\}_{K_B}$ and $\{\!|B\ ,\ K\ ,\ N|\!\}_{K_A}$ (for some N). Since in addition, by **KS1**, $K \notin \mathsf{K}_{\mathcal{P}}$, it follows that $K \in \mathsf{S}_1$, and is safe.

Thus $+N_b \Rightarrow -\{\!|N_b|\!\}_K$ is also an incoming test, and guarantees a transforming edge $N_b \rightsquigarrow \{\!|N_b|\!\}_K$. It can lie only on an initiator strand $s_i \in \mathrm{Init}[I, R, N, N_b, K]$. We would like to ensure that I and R are A and B respectively. By the form of the strand, we know that $\{\!|R\ ,\ K\ ,\ N|\!\}_{K_I} \sqsubset \mathrm{term}(s_i \downarrow 2)$. However, K is transmitted only in the forms $\{\!|A\ ,\ B\ ,\ K\ ,\ N_b|\!\}_{K_B}$ and $\{\!|B\ ,\ K\ ,\ N|\!\}_{K_A}$. Since the first term does not match, $\{\!|R\ ,\ K\ ,\ N|\!\}_{K_I} = \{\!|B\ ,\ K\ ,\ N|\!\}_{K_A}$, so $R = B$ and $I = B$.

We need not worry about key compromise in this protocol. A's incoming test guarantees that S generates K after receiving N_a. So the key is more recent than the beginning of A's own strand. This strand will be implemented to time out long before any cryptanalytic attack could be completed, so K cannot have been compromised yet. The same reasoning applies to B and N_b.

4.2 The Weakened Yahalom Protocol

The important change in the Yahalom-Paulson protocol was to include the responder's nonce N_b as part of the encrypted ticket $\{\!|A\ ,\ B\ ,\ K\ ,\ N_b|\!\}_{K_B}$ generated by the server. In contrast, a weakened Yahalom protocol (also discussed by Paulson) omits this nonce from the encrypted ticket. It is transmitted in plaintext and returned encrypted under the session key K, but not in the ticket (Figure 5). An authentication test analysis also verifies the correctness of this protocol, ignoring the risk of key compromise.

No compromise Focusing on responder's point of view, we cannot start with an incoming test as in the Yahalom-Paulson protocol, because N_b is not

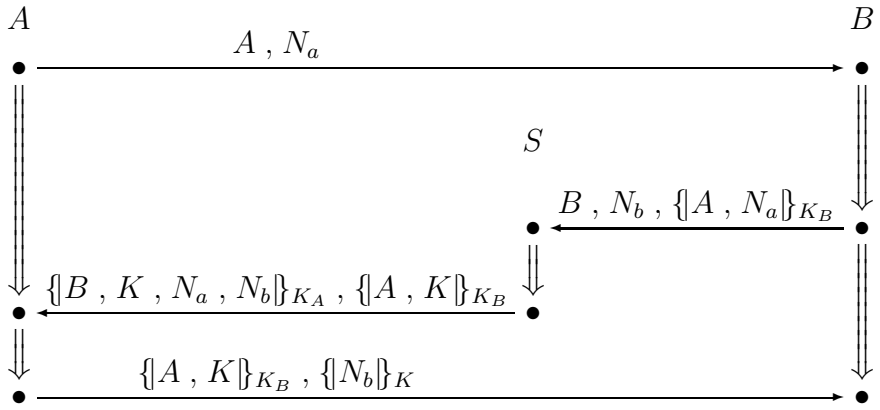


Fig. 5. The Weakened Yahalom Protocol

returned encrypted under K_B, and we do not yet know whether $K \in \mathsf{S}$.

Instead, we start with an unsolicited test. $\{\!|A , K|\!\}_{K_B}$ is encrypted with a safe key, so a regular node originates $\rightsquigarrow \{\!|A , K|\!\}_{K_B}$. By the form of the protocol, this positive node is on $s_s \in \mathrm{Serv}[A, B, *, *, K]$. By **KS1** and **KS2**, the only transmissions of K are protected by K_A and K_B, so K is safe.

We may now infer that the edge $+ \ldots N_b \cdots \Rightarrow -\{\!|N_b|\!\}_K$ is an incoming test, guaranteeing a regular transforming edge $N_b \rightsquigarrow \{\!|N_b|\!\}_K$. This is $s_i \downarrow 2 \Rightarrow s_i \downarrow 3$ for $s_i \in \mathrm{Init}[I, R, *, N_b, K]$. However, by an unsolicited test, there exists $s'_s \in \mathrm{Serv}[I, R, *, *, K]$, so by **KS2** $s_s = s'_s$, whence $I = A$ and $R = B$.

Compromised by time Key compromise changes this situation. Because we started with an unsolicited test, we do not know how long ago the server strand s_s occurred. In particular, it may have been before a previous session in which the key K encrypted a large amount of data. Cryptanalysis may thus have disclosed it. If the penetrator has access to K, then the edge $\ldots N_b \cdots \Rightarrow \{\!|N_b|\!\}_K$ is no longer an incoming test, invalidating the rest of our argument.

Figure 6 illustrates what might go wrong. In the upper left hand corner, we show incompletely an old server strand marked S. The term t_B stands for the ticket $\{\!|A , K|\!\}_{K_B}$, and t_A stands for the corresponding encrypted message for A. K originates in the center of Figure 6 as a result of cryptanalysis, which we have no desire to represent more explicitly. A , N_a is a message sent by the penetrator, purporting to be A. Separation strands are shown in incomplete form when one result is discarded.

This key compromise attack is perhaps less obvious than the NSSK attack, since there is no suspicious neck in the protocol definition. However, the underlying problem is the same: B cannot verify the recency of the session key K. And the cause of the problem is the same, namely that the protocol relies on an unsolicited test rather than an incoming or outgoing test.

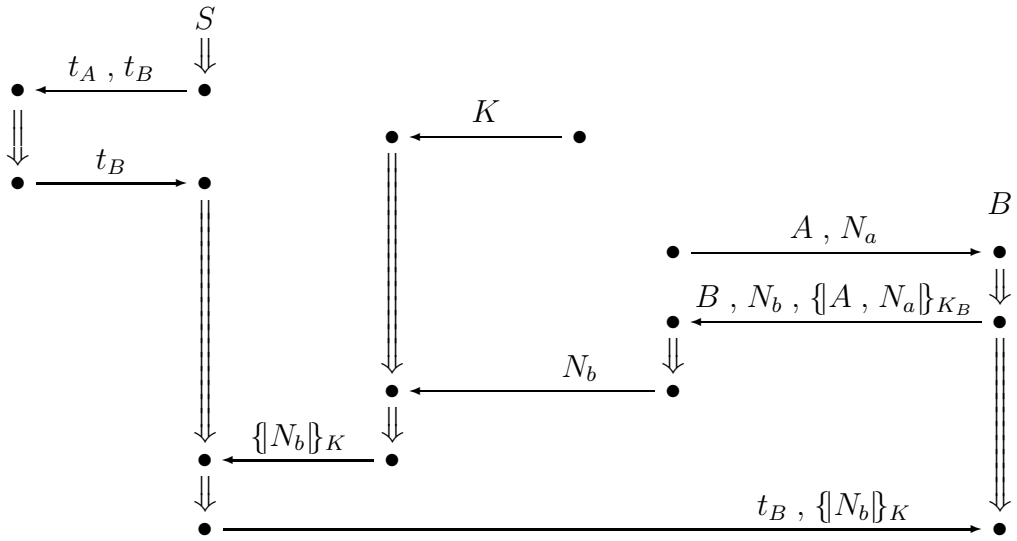


Fig. 6. Key Compromise Attack on Weakened Yahalom

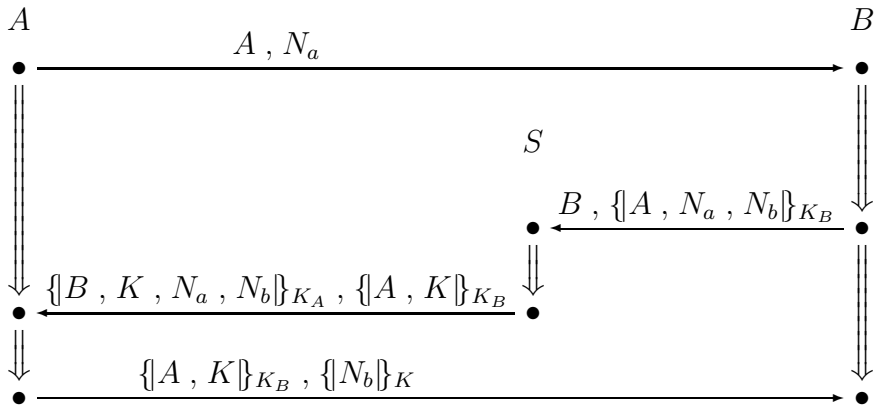


Fig. 7. The Yahalom Protocol

4.3 The Yahalom Protocol

The original Yahalom protocol (Figure 7) lies in between the Yahalom-Paulson protocol and the weakened version. It avoids sending the nonce N_b in plaintext, yet it does not include the nonce in the ticket S generates for B. Our question is, does this save the protocol from the key-compromise attack? To answer this question, we will extract a notion of recency from the examples, and use it to provide a more accurate specification for key servers.

5 Recency

We first define a useful notion of recency motivated by the causal ordering relation on bundles. We then adapt the constraints **KS1**–**KS2** on key servers,

so that they entail that session keys are uncompromised *if recently generated*.

5.1 A Notion of Recency

Our core idea is that regular strands provide a way to measure recency. Implementors always ensure that a protocol run will time out long before cryptanalysis could have succeeded, a matter of hours at least in the case of any usable cryptosystem. [1] Thus, a principal engaged in a strand knows that an event is recent, if it happened after an earlier event on the same strand.

Definition 5.1 (**Recency for**) A node n is *recent for* a regular node m_1 in $\mathcal{B}$ if there is a regular node $m_0 \in \mathcal{B}$ such that $m_0 \Rightarrow^+ m_1$ and $m_0 \preceq_{\mathcal{B}} n \prec_{\mathcal{B}} m_1$.

The incoming and outgoing tests entail recency. That is, if $m_0 \Rightarrow^+ m_1$ is a test edge, and $n_0 \Rightarrow n_1$ is the corresponding transforming edge in $\mathcal{B}$, then $m_0 \preceq n_0 \prec n_1 \prec m_1$, so that n_0 and n_1 are recent for m_1. By contrast, an unsolicited test provides no evidence of recency.

5.2 Key Servers and Recency

We now reformulate two of our principles governing key servers. The third, **KS3**, does not need reformulation, since it stipulates that session keys are disjoint from long term keys.

The other two original principles **KS1**–**KS2** expressed the assumption that a session key would be uncompromised forever. The more realistic assumption is that it is uncompromised if recently generated. There are two ingredients in this, one being that the penetrator will not guess a recent session key. The other ingredient is that if a key server generated a session key K recently before a node n, then no other server run that could affect n also generated the same key K. We combine these two ingredients in a single principle.

We say that K *originates uniquely previous to* m in $\mathcal{B}$ if there exists a unique node n such that $n \preceq_{\mathcal{B}} m$ and K originates at n. The nodes previous to m are all those that could have a causal influence on m, its backward light cone, to borrow a metaphor from the theory of relativity.

RKS If $\exists s \,.\, s \in \mathrm{Serv}[**, K]$ and some node $s \downarrow i$ on s is recent for m in $\mathcal{B}$, then K originates uniquely previous to m in $\mathcal{B}$.

In reasonable key distribution protocols, K originates on s if $s \in \mathrm{Serv}[**, K]$. Thus, **RKS** entails that there is no penetrator K-node previous to m emitting the session key K. That would be another originating node. It also entails that no other server strand emits the same session key K previous to m. Thus, it covers the intended effect of **KS1**–**KS2**, in case the server run was recent, for the part of $\mathcal{B}$ that can affect m.

[1] Also, cryptanalysis can typically begin only after a key establishment protocol has completed, as the traffic from an encrypted session is a more promising source than the protocol messages themselves.

Sub-Bundles How can we reason about the nodes previous to a given node m? If $\mathcal{B}$ is a bundle and $m \in \mathcal{B}$ is any node in it, then the set of nodes $S = \{n \in \mathcal{B} \colon n \preceq_{\mathcal{B}} m\}$ is a sub-bundle $\mathcal{B}_0$, when equipped with those edges of $\mathcal{B}$ both endpoints of which are in S [6].

Suppose moreover that we are trying to establish an authentication result for a particular strand s in $\mathcal{B}$. The logical form of authentication results ([19], cp. [11,20]) requires us to show that a corresponding portion of another strand s' is also included in $\mathcal{B}$. Let m be the last node on s such that $m \in \mathcal{B}$, and consider the sub-bundle $\mathcal{B}_0$ of nodes $n \preceq_{\mathcal{B}} m$. If we can prove that the required portion of s' is included in the sub-bundle $\mathcal{B}_0$, then it is also included in $\mathcal{B}$.

Thus, when proving an authentication result, we may ignore the portion of $\mathcal{B}$ not previous to the nodes of interest. We then reason about the sub-bundle, and transfer the authentication results back to $\mathcal{B}$. The transfer is valid even if the conclusions are drawn using the authentication test method using the safe keys S_0 and S_1 within $\mathcal{B}_0$, i.e. $\mathsf{S}_0(\mathcal{B}_0)$ and $\mathsf{S}_1(\mathcal{B}_0)$.

5.3 Analysis of Yahalom

We now show why Yahalom's protocol is correct, even when non-recent session keys may be compromised. We again focus on the responder.

Suppose that a bundle $\mathcal{B}$ contains all three nodes of a regular strand $s_r \in \mathrm{Resp}[A, B, N_a, N_b, K]$, and N_b is uniquely originating. What authentication guarantees does the responder B have about the other participants in $\mathcal{B}$? Let $\mathcal{B}_0$ be the sub-bundle of $\mathcal{B}$ containing the nodes $n \in \mathcal{B}$, such that $n \preceq_{\mathcal{B}} (s_r \downarrow 3)$.

First, since $K_B \in \mathsf{S}_0$, the edge $s_r \downarrow 2 \Rightarrow s_r \downarrow 3$ is an outgoing test with test component $\{\!|A\ ,\ N_a\ ,\ N_b|\!\}_{K_B}$. Thus, there is a recent transforming edge $\{\!|A\ ,\ N_a\ ,\ N_b|\!\}_{K_B} \rightsquigarrow N_b$. The latter must lie on a recent server strand $s_s \in \mathrm{Serv}[A, B, N_a, N_b, K']$ for some K'.

By **RKS**, we may infer that K' originates uniquely previous to $S_r \downarrow 3$, and thus originates uniquely in $\mathcal{B}_0$. In particular, $K' \in \mathsf{S}$. Since K' does not occur in the test term $\{\!|A\ ,\ N_a\ ,\ N_b|\!\}_{K_B}$, we cannot yet say whether $K' = K$.

We now use the extension to the outgoing test condition mentioned above in Section 2.3 and justified in [7, Section 4.2.1]. N_b is again transmitted on $s_s \downarrow 2$, contained only in the form $\{\!|B\ ,\ K'\ ,\ N_a\ ,\ N_b|\!\}_{K_A}$. Since $K_A \in \mathsf{S}$, it implies that $\{\!|B\ ,\ K'\ ,\ N_a\ ,\ N_b|\!\}_{K_A}$ is also a subterm of a regular receiving node.[2] Considering the cases possible in this protocol, that node is $s_i \downarrow 2$ where $s_i \in \mathrm{Init}[A, B, N_a, N_b, K']$. Thus, $\mathrm{term}(s_i \downarrow 3) = \{\!|N_b|\!\}_{K'}$.

Since $K' \in \mathsf{S}$, we may apply the same principle again, inferring that $\{\!|N_b|\!\}_{K'}$ is also a subterm of a regular receiving node. In the protocol, that node is $s'_r \downarrow 3$ where $s'_r \in \mathrm{Resp}[**, N_b, K']$. However, since N_b is uniquely originating, and originates on both s_r and s'_r, $s_r = s'_r$. Therefore, $K = K'$.

[2] Strictly speaking, the relevant fact is $K_A^{-1} \in \mathsf{S}$, but $K_A^{-1} = K_A$ because the protocols considered in this paper use only symmetrical keys (Assumption 2).

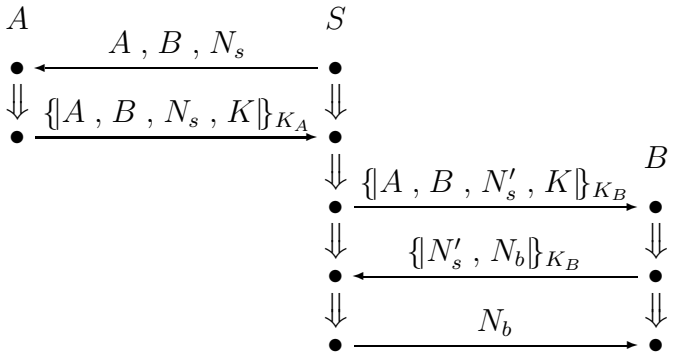


Fig. 8. The Trusted Introducer Protocol

Thus, $s_s \in \mathrm{Serv}[A, B, N_a, N_b, K]$ and $s_i \in \mathrm{Init}[A, B, N_a, N_b, K]$. This establishes B's authentication guarantees in $\mathcal{B}_0$. By our transfer principle for authentication results, the same guarantees also hold in the larger bundle $\mathcal{B}$. Secrecy goals are of course a different matter. The secrecy of the session key K is guaranteed in $\mathcal{B}_0$, as a consequence of **RKS**. However, nothing guarantees that K remains secret in $\mathcal{B}$, because $\mathcal{B}$ may contain nodes for which s_s is no longer recent. Those nodes may issue in a disclosure of K, for instance as a consequence of cryptanalysis.

6 Conclusion

In this paper, we have distinguished between protocols like NSSK or Weakened Yahalom, which are susceptible to key compromise attacks, and those like Otway-Rees [1,14,19] or Yahalom, which are not. Our notion of recency is perfectly adapted to our protocol verification method, the authentication test method: the incoming and outgoing authentication tests provide a guarantee of the recency of their transforming edges.

This notion of recency is not complete. In the slightly peculiar protocol shown in Figure 8, the principals share a long term key with a server, which is only a trusted introducer; it does not generate session keys. The initiator furnishes the session key. The responder uses an outgoing test to ensure a recent server's session $s_s \in \mathrm{Serv}[A, B, *, N'_s, N_b, K]$. The server had used an incoming test to ensure an initiator's session $s_i \in \mathrm{Init}[A, B, N_s, K]$ recent for s_s. However, s_i is not recent for s_r, although it was *recently* recent.

We want a sort of "extension ladder" notion of recency here. Let us define:

(i) A node n is 1-recent for m if n is recent for m as in Definition 5.1;

(ii) A node n is $i+1$-recent for m_1 if there exists a node m_0 such that n is i-recent for m_0 and m_0 is recent for m_1.

If n is i-recent for m, then there are $i+1$ strands, each overlapping a portion of the preceding one. From beginning to end, at most $i+1$ times the time-

out for a single regular strand can have elapsed. A particular cryptosystem and protocol implementation determines what values of i are small enough to avoid key compromise. The protocol determines what degree of recency is guaranteed, which for the introducer's protocol is 2-recency.

Acknowledgments Supported by the National Security Agency under US Army CECOM contract number DAAB07-99-C-C201.

Larry Paulson suggested the Yahalom protocol to me as a provocative test case. Gavin Lowe described related work of his own to me, and suggested the "extension ladder" notion of recency. They also made helpful comments on a previous draft. I am grateful to the two of them, and also to Javier Thayer, with whom I discussed all this.

References

[1] Burrows, M., M. Abadi and R. Needham, *A logic of authentication*, Proceedings of the Royal Society **Series A, 426** (1989), pp. 233–271, 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] Denning, D. and G. Sacco, *Timestamps in key distribution protocols*, Communications of the ACM **24** (1981).

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

[4] Guttman, J. D., *Security goals: Packet trajectories and strand spaces*, in: R. Gorrieri and R. Focardi, editors, *Foundations of Security Analysis and Design*, LNCS **2171**, Springer Verlag, 2001 Forthcoming.

[5] Guttman, J. D. and F. J. Thayer Fábrega, *Authentication tests*, in: *Proceedings, 2000 IEEE Symposium on Security and Privacy*, May (2000).

[6] Guttman, J. D. and F. J. Thayer Fábrega, *Protocol independence through disjoint encryption*, in: *Proceedings, 13th Computer Security Foundations Workshop* (2000).

[7] Guttman, J. D. and F. J. Thayer Fábrega, *Authentication tests and the structure of bundles*, Theoretical Computer Science (2001), to appear.

[8] Heather, J. and S. Schneider, *Toward automatic verification of authentication protocols on an unbounded network*, in: *Proceedings, 13th Computer Security Foundations Workshop* (2000).

[9] Lowe, G., *Breaking and fixing the Needham-Schroeder public-key protocol using FDR*, in: *Proceeedings of* tacas, Lecture Notes in Computer Science **1055** (1996), pp. 147–166.

[10] Lowe, G., *Casper: A compiler for the analysis of security protocols*, in: *10th Computer Security Foundations Workshop Proceedings* (1997), pp. 18–30.

[11] Lowe, G., *A hierarchy of authentication specifications*, in: *10th Computer Security Foundations Workshop Proceedings* (1997), pp. 31–43.

[12] Marrero, W., 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, 1997.

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

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

[15] Paulson, L. C., *The inductive approach to verifying cryptographic protocols*, Journal of Computer Security (1998), also Report 443, Cambridge University Computer Lab.

[16] Paulson, L. C., *Relations between secrets: Two formal analyses of the Yahalom protocol*, Journal of Computer Security (2001), also available as Cambridge University Computer Laboratory Technical Report 432 (1997).

[17] Perrig, A. and D. X. Song, *Looking for diamonds in the desert: Extending automatic protocol generation to three-party authentication and key agreement protocols*, in: *Proceedings of the 13th IEEE Computer Security Foundations Workshop* (2000).

[18] Schneider, S., *Verifying authentication protocols with CSP*, in: *Proceedings of the 10th IEEE Computer Security Foundations Workshop* (1997), pp. 3–17.

[19] Thayer Fábrega, F. J., J. C. Herzog and J. D. Guttman, *Strand spaces: Proving security protocols correct*, Journal of Computer Security **7** (1999), pp. 191–230.

[20] Woo, T. Y. C. and S. S. Lam, *Verifying authentication protocols: Methodology and example*, in: *Proc. Int. Conference on Network Protocols*, 1993.

A Strand Space Definitions

This appendix, derived from [4,7,19], defines the basic strand space notions.

A.1 Strands, Strand Spaces, and Origination

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 with a trace mapping $tr : \Sigma \to (\pm\mathsf{A})^*$.

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.

(i) A *node* is a pair $\langle s, i \rangle$, with $s \in \Sigma$ and i an integer satisfying $1 \leq i \leq \text{length}(\text{tr}(s))$. 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.

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

(iii) There is an edge $n_1 \to n_2$ if and only if $\text{term}(n_1) = +a$ and $\text{term}(n_2) = -a$ 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.

(iv) 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.

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

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

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

(viii) An unsigned term t is *uniquely originating in* a set of nodes $S \subset \mathcal{N}$ iff there is a unique $n \in S$ such that t originates on n.

(ix) An unsigned term t is *non-originating in* a set of nodes $S \subset \mathcal{N}$ iff there is no $n \in S$ such that t originates on n.

If a term t originates uniquely in a suitable set of nodes, then it can play the role of a nonce or session key, assuming that everything that the penetrator does in some scenario is in that set of nodes.

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

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:

(i) $\mathcal{N}_{\mathcal{C}}$ and $\rightarrow_{\mathcal{C}} \cup \Rightarrow_{\mathcal{C}}$ are finite.

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

(iii) If $n_2 \in \mathcal{N}_{\mathcal{C}}$ and $n_1 \Rightarrow n_2$ then $n_1 \Rightarrow_{\mathcal{C}} n_2$.

(iv) $\mathcal{C}$ is acyclic.

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

Definition 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*$(s) = \langle \mathrm{tr}(s)(1), \ldots, \mathrm{tr}(s)(m) \rangle$, where $m = \mathcal{C}$-height(s).

We say that $s \in \mathcal{C}$ if the $\mathcal{C}$-height of s equals length(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 $\mathcal{S}$.

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 $\mathbf{inv} : \mathsf{K} \rightarrow \mathsf{K}$. We assume that $\mathbf{inv}$ 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 $\mathbf{encr} : \mathsf{K} \times \mathsf{A} \rightarrow \mathsf{A}$ and $\mathbf{join} : \mathsf{A} \times \mathsf{A} \rightarrow \mathsf{A}$.

We follow custom and write $\mathbf{inv}(K)$ as K^{-1}, $\mathbf{encr}(K, m)$ as $\{|m|\}_K$, and $\mathbf{join}(a, b)$ as $a\ ,\ b$. In this paper, we are concerned only with symmetric-key protocols, so we assume that $K = K^{-1}$ always.

We assume, like many others (e.g. [10,12,15]), that A is freely generated.

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.

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.8 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.

Since in this paper we assume $K = K^{-1}$, the key used in a D-strand is the same as the one used to create the ciphertext.