A Bound on Attacks on Payment Protocols

Scott D. Stoller*
Computer Science Dept., SUNY at Stony Brook, Stony Brook, NY 11794-4400 USA

April 10, 2001

Abstract

Electronic payment protocols are designed to work correctly in the presence of an adversary that can prompt honest principals to engage in an unbounded number of concurrent instances of the protocol. This paper establishes an upper bound on the number of protocol instances needed to attack a large class of protocols, which contains versions of some well-known electronic payment protocols, including SET and 1KP. Such bounds clarify the nature of attacks on and provide a rigorous basis for automated verification of payment protocols.

1 Introduction

Many protocols, including electronic payment protocols, are designed to work correctly in the presence of an adversary (also called a penetrator) that can prompt honest principals to engage in an unbounded number of concurrent instances of the protocol. Payment protocols should satisfy at least two kinds of correctness requirements: *secrecy*, which states that certain values are not obtained by the penetrator, and *agreement*, which states that a principal executes a certain action only if appropriate other principals previously executed corresponding other actions (*e.g.*, a payment gateway approves a charge to customer C's account only if customer C previously authorized that charge).

Allowing an unbounded number of concurrent protocol instances makes the number of reachable states unbounded. The case studies in, *e.g.*, [MCF87, Ros95, HTWW96, DK97, LR97, MMS97, MCJ97, MSS98, Bol98, DNL99, MM99] show that state-space exploration of security protocols is feasible when small upper bounds are imposed on the size of messages and the number of protocol instances. In most of those case studies, the bounds are not rigorously justified, so the results do not prove correctness of the protocols. Rigorous automated verification of these protocols requires either symbolic state-space exploration algorithms that directly accommodate these infinite state spaces or theorems that reduce correctness of these protocols to finite-state problems.

This paper presents a reduction for a large class of protocols. It uses the strand space model [THG98]. A regular strand can be regarded as a thread that runs the program corresponding to one role of the protocol and then terminates; thus, a regular strand corresponds to one instance of one role in the protocol. A central hypothesis of our reduction is the bounded support restriction (BSR), which states that in every history (*i.e.*, every possible behavior) of the protocol, each regular strand depends on at most a given number of other regular strands. Our notion of dependence, embodied in the definition of *support*, is a variant of Lamport's happened-before relation [Lam78], modified to handle freshness of nonces appropriately. BSR is not easily checked by static analysis, so we propose to check it by state-space exploration, while checking the correctness requirements. With statically checkable restrictions alone, it seems difficult to find restrictions that are both strong enough to justify a reduction and weak enough to be satisfied by well-known protocols.

*The author gratefully acknowledges the support of NSF under Grant CCR-9876058 and the support of ONR under Grants N00014-99-1-0358 and N00014-01-1-0109. Email: stoller@cs.sunysb.edu Web: http://www.cs.sunysb.edu/~stoller/ Phone: 631-632-1627

To check BSR by state-space exploration, we need a reduction for it. We prove: if a protocol satisfies its correctness requirements and BSR when appropriate bounds are imposed on the number of regular strands in a history, then the protocol also satisfies its correctness requirements and BSR without those bounds. We actually prove the contrapositive of this statement, by supposing that some history of the protocol violates BSR or a correctness requirement and constructing a history violating the same property and containing a bounded number of regular strands. That history is constructed by starting from an earliest node (in the strand space model, events are usually called "nodes") that causes a violation of the property and finding the set of nodes on which that node depends. Roughly speaking, that set of nodes, augmented with appropriate actions by the adversary, is the desired history.

Most existing techniques for automated analysis of systems with unbounded numbers of concurrent processes, such as [CGJ95, KM95, EN96, AJ98, BSBL00, JN00], are not applicable to payment protocols, because they assume the set of values (equivalently, the set of local states of each process) is independent of the number of processes, whereas payment protocols generate fresh values, so the set of values grows as the number of processes (strands) increases. Dolev and Yao's algorithms for verifying secrecy requirements of cryptographic protocols are interesting but limited [DY83]; they do not handle agreement requirements, and they apply to a severely restricted class of protocols, which excludes the variants of SET and 1KP described in Section 2.1 and is strictly included in the class of protocols handled by our reduction.

Roscoe and Broadfoot use data-independence techniques to bound the number of nonces needed for an attack [RB99]. Their result assumes that each trustworthy principal participates in at most a given number of protocol instances at a time. Our reduction does not require that assumption; indeed, our goal is to justify such assumptions. Lowe's reduction [Low99] has similar goals as our reduction and provides tighter bounds in its domain of applicability, but it does not handle agreement requirements and does not apply to the variants of SET and 1KP described in Section 2.1.

The reduction embodied in Theorems 2 and 3 handles secrecy and agreement requirements and applies to simplified versions of SET [SET97] and 1KP [BGH+00]. It extends the reduction in [Sto99] in several significant ways. The class of preserved properties is extended to allow protocol-specific secrecy properties (roughly, any non-cryptographic value can be designated as a secret) and to allow use of more general predicates to characterize the desired relationship between actions in agreement properties. The class of protocols is extended by allowing hash functions, allowing arbitrary nesting of hashing and encryption in protocol messages, and relaxing the restriction that the recipient of a message be able to recognize the entire structure of the message.[1] These extensions necessitate substantial changes to the statement and proof of Theorem 1. That theorem is the crux of the proof of our reduction: it provides a statically-calculated bound on a "dynamic" quantity (*i.e.*, a quantity defined by a maximum over all possible executions of the protocol); that quantity is the dependence width, defined in Section 4.

Our results implicitly describe a simulation relation between systems with bounded-size histories and systems with unbounded-size histories. It would be interesting to see whether similar results could be obtained more easily in a process-algebraic framework, such as Spi calculus [AG99].

2 Model of Protocols

We use the strand space model [THG98], with minor modifications. We introduce simple languages for protocols and correctness requirements.

The set of *primitive terms* is $Prim = Text \cup Key$, where $Text$ is a set of values other than cryptographic keys, and $Key = \{key(x, y) \mid x, y \in Name \wedge x \neq y\} \cup \{pub(x) \mid x \in Name\} \cup \{pvt(x) \mid x \in Name\}$. Informally,

[1] Session keys are not used in the examples in this paper, so we omitted them from the framework. They can be handled roughly as in [Sto99].

$key(x, y)$ is a symmetric key intended for use by x and y, and $pub(x)$ and $pvt(x)$ represent x's public and private keys, respectively, in a public-key cryptosystem. *Name* is the subset of *Text* containing names of principals. *Nonce* is the subset of *Text* containing nonces.

The set *Term* of terms is defined inductively as follows. (1) All primitive terms are terms. (2) If t and t' are terms and $k \in Key$, then $encr(t, k)$ (encryption of t with k, usually written $\{t\}_k$), $pair(t, t')$ (pairing of t and t', usually written $t \cdot t'$), and $h(t)$ (hash of t, where h represents a one-way collision-resistant hash function [MvOV97]) are terms.

$\text{inv} \in Key \to Key$ maps each key to its inverse: decrypting $\{t\}_k$ with $\text{inv}(k)$ yields t. For a symmetric key k, $\text{inv}(k) = k$. We usually write $\text{inv}(k)$ as k^{-1}. We assume perfect encryption, *i.e.*, $\{t\}_k = \{t'\}_{k'}$ iff $t = t'$ and $k = k'$. Distinct primitive terms are assumed to represent distinct values (*e.g.*, $key(A, B)$ and $key(A, S)$ represent different keys).

$[t]_{pvt(x)}$ abbreviates $t \cdot \{h(t)\}_{pvt(x)}$, *i.e.*, t signed by x.

A *ciphertext* is a term whose outermost operator is *encr*. A *hash* is a term whose outermost operator is h. A term t' *occurs in the clear* in t if there is an occurrence of t' in t that is not in the scope of *encr* or h. For example, in the term $\{A\}_{k_1} \cdot \{\{B\}_{k_1}\}_{k_2}$, the term $\{A\}_{k_1}$ occurs in the clear, and the term $\{B\}_{k_1}$ does not.

Let $\text{dom}(f)$ denote the domain of a function f. A sequence is a function whose domain is a finite prefix of the natural numbers. Let $\text{len}(\sigma)$ denote the length of a sequence σ. $\langle\!\langle a, b, \ldots \rangle\!\rangle$ denotes a sequence σ with $\sigma(0) = a$, $\sigma(1) = b$, and so on.

A *directed term* is $+t$ or $-t$, where t is a term. Positive and negative terms represent sending and receiving messages, respectively. We sometimes refer to directed terms as "terms" and treat them as terms, for instance as having subterms.

A *trace* is a finite sequence of directed terms. Let *Trace* denote the set of traces.

A *trace mapping* is a function $tr \in \text{dom}(tr) \to Trace$, where $\text{dom}(tr)$ is an arbitrary set whose elements are called *strands*.

A *node* of tr is a pair $\langle s, i \rangle$ with $s \in \text{dom}(tr)$ and $0 \leq i < \text{len}(tr(s))$. Let $\mathcal{N}_{tr}$ denote the set of nodes of tr. We say that node $\langle s, i \rangle$ is on strand s. Let $\text{nodes}_{tr}(s)$ denote the set of nodes on strand s in tr. Let $\text{strand}(\langle s, i \rangle) = s$, $\text{index}(\langle s, i \rangle) = i$, and $\text{term}_{tr}(\langle s, i \rangle) = tr(s)(i)$. If $\text{term}_{tr}(n)$ is positive (or negative), we say that n is positive (or negative).

The local dependence relation is: $\langle s_1, i_1 \rangle \stackrel{lcl}{\to} \langle s_2, i_2 \rangle$ iff $s_1 = s_2$ and $i_2 = i_1 + 1$.

A term t *originates* from a node $\langle s, i \rangle$ in tr iff $\langle s, i \rangle$ is positive, t is a subterm of $\text{term}_{tr}(\langle s, i \rangle)$,[2] and t is not a subterm of $\text{term}_{tr}(\langle s, j \rangle)$ for any $j < i$.

A term t *uniquely originates* from a node n in tr iff it originates from n in tr and not from any other node in tr. Typically, nonces are uniquely-originated. This is the strand space way of expressing freshness.

For $S \subseteq \mathcal{N}_{tr}$, let $\text{term}_{tr}(S) = \{\text{term}_{tr}(n) \mid n \in S\}$. For symbols subscripted by the trace mapping, we elide the subscript when the trace mapping is evident from context.

2.1 Roles, Protocols, and Penetrator

A *role* is a parameterized sequence of directed terms. Associated with each parameter is a type, *i.e.*, a set of allowed terms. Some parameters with type *Nonce* may be designated as uniquely-originated; informally, this means that the value of that parameter must be uniquely-originated. Uniquely-originated parameters are designated by underlining in the parameter list. We require that for every role r, for every parameter x of r with type *Nonce*, x is uniquely-originated iff the first occurrence of x in r is in a positive term. Let $r.x$ denote

[2] We use the standard notion of subterm, rather than the modified subterm relation $\sqsubseteq$ defined in [THG98], in which k is not necessarily a subterm of $\{t\}_k$. Our definition induces a stronger notion of uniquely-originates. This difference is inessential.

parameter x of role r. For example, $R(\underline{nc} : Nonce) = \langle\!\langle +nc \rangle\!\rangle$ defines a role R with one uniquely-originated parameter nc with type *Nonce*.

A *trace for role* r is a prefix of a trace obtained by substituting for each parameter x of r a term in the type of x. The requirement that parameters be instantiated with values of the specified types is sometimes called the *strong typing assumption*. This assumption is common in protocol analysis, but ensuring that it provides a reasonable abstraction of a given implementation is non-trivial. A role r and a trace σ for r uniquely determine a mapping, denoted $args(r, \sigma)$, from the set of parameters of r that appear in $r(0), r(1), \ldots, r(\text{len}(\sigma) - 1)$ to *Term*. For example, for role $R(x_1 : Name, x_2 : Name) = \langle\!\langle +x_1, +x_2 \rangle\!\rangle$ and $\sigma = \langle\!\langle +A \rangle\!\rangle$, $\text{dom}(args(R, \sigma)) = \{x_1\}$ and $args(R, \sigma)(x_1) = A$.

A *protocol* Π is a set of roles, together with a set $\Pi.Secret \subseteq (Text \setminus (Name \cup Nonce))$ of terms that are "secrets" (*i.e.*, terms that should not be revealed to the penetrator). Excluding names here implies that the penetrator knows all names. Specialized notions of secrecy are used for keys and nonces, as described in Section 2.5.

The penetrator model is parameterized by a set $Key_P \subset Key$ of keys initially known to the penetrator. The set $\Pi_P(Key_P)$ of *penetrator roles* contains:

Pair: $P(x_1 : Term, x_2 : Term) = \langle\!\langle -x_1, -x_2, +x_1 \cdot x_2 \rangle\!\rangle$	Message: $M(x : Text \cup Nonce) = \langle\!\langle +x \rangle\!\rangle$
Separation: $S(x_1 : Term, x_2 : Term) = \langle\!\langle -x_1 \cdot x_2, +x_1, +x_2 \rangle\!\rangle$	Key: $K(k : Key_P) = \langle\!\langle +k \rangle\!\rangle$
Encryption: $E(k : Key, x : Term) = \langle\!\langle -k, -x, +\{x\}_k \rangle\!\rangle$	Hash: $H(x : Term) = \langle\!\langle -x, +h(x) \rangle\!\rangle$
Decryption: $D(k : Key, x : Term) = \langle\!\langle -k^{-1}, -\{x\}_k, +x \rangle\!\rangle$	

Typically, $Key_P = \{key(x, y) \in Key \mid x = P \vee y = P\} \cup \{pvtkey(P)\} \cup \{pubkey(x) \mid x \in Name\}$.

2.2 History

A *history of protocol* Π is a tuple $h = \langle tr, \overset{msg}{\rightarrow}, role \rangle$, where tr is a trace mapping, $\overset{msg}{\rightarrow}$ is a binary relation on $\mathcal{N}_{tr}$, and $role \in \text{dom}(tr) \to (\Pi \cup \Pi_P(Key_P))$ such that

1. For all $n_1, n_2 \in \mathcal{N}_{tr}$, if $n_1 \overset{msg}{\rightarrow} n_2$, then there exists $t \in Term$ such that $\text{term}_{tr}(n_1) = +t$ and $\text{term}_{tr}(n_2) = -t$. This represents that n_1 sends t, and n_2 receives t.
2. For all $n_1 \in \mathcal{N}_{tr}$, if $\text{term}_{tr}(n_1)$ is negative, then there exists exactly one $n_2 \in \mathcal{N}_{tr}$ such that $n_2 \overset{msg}{\rightarrow} n_1$.
3. $\preceq_h$ is acyclic and well-founded (*i.e.*, does not have infinite descending chains), where $\preceq_h$ is the reflexive and transitive closure of $(\overset{msg}{\rightarrow} \cup \overset{lcl}{\rightarrow})$. Note that $\preceq_h$ is a partial order, first defined by Lamport [Lam78].
4. For all $s \in \text{dom}(tr)$, $tr(s)$ is a trace for $role(s)$. A *regular strand* is a strand s with $role(s) \in \Pi$. A *penetrator strand* is a strand s with $role(s) \in \Pi_P(Key_P)$. Nodes on regular and penetrator strands are called *regular nodes* and *penetrator nodes*, respectively. (For convenience, we assume $\Pi \cap \Pi_P(Key_P) = \emptyset$.)
5. For all $s \in \text{dom}(tr)$, for all $x \in \text{dom}(args(role(s), tr(s)))$, if parameter x is uniquely-originated, then $args(role(s), tr(s))(x)$ uniquely originates from $\langle s, i \rangle$, where i is the index of the first term in r that contains x.
6. For all $t \in \Pi.Secret$, t originates only from regular nodes.

Note that a history may contain multiple traces for the same role with identical bindings for parameters that are not uniquely originated.

To reduce clutter, we sometimes use a history instead of a trace mapping as a subscript; *e.g.*, for a history $h = \langle tr, \overset{msg}{\rightarrow}, role \rangle$, we define $\mathcal{N}_h = \mathcal{N}_{tr}$.

The set of *predecessors* of a node n in a history h is $\text{preds}_h(n) = \{n' \in \mathcal{N}_h \mid n' \preceq_h n \ \wedge \ n' \neq n\}$.

Cust($c : Name_c$, $m : Name_m \cup \{P\}$, $g : Name_g \cup \{P\}$, $\underline{nc} : Nonce$, $nm : Nonce$,
$price : Price$, $od : Order$, $pd : PayDesc$, $result : Result$) =
let $trans = c \cdot m \cdot g \cdot nc \cdot nm \cdot price \cdot h(od) \cdot h(pd)$ **in**
$\langle\!\langle +c \cdot m,$ (* 1. to merchant *)
$-nm,$ (* 2. from merchant *)
$+[trans]_{pvt(c)} \cdot \{od\}_{pub(m)} \cdot \{pd\}_{pub(g)},$ (* 3. to merchant *)
$-[result \cdot h(trans)]_{pvt(g)} \rangle\!\rangle$ (* 4. from gateway *)

Mrch($c : Name_c \cup \{P\}$, $m : Name_m$, $g : Name_g \cup \{P\}$, $nc : Nonce$, $\underline{nm} : Nonce$,
$price : Price$, $od : Order$, $hpd : Hash(PayDesc)$, $epd : Term$, $result : Result$) =
let $trans = c \cdot m \cdot g \cdot nc \cdot nm \cdot price \cdot h(od) \cdot hpd$ **in**
$\langle\!\langle -c \cdot m,$ (* 1. from customer *)
$+nm,$ (* 2. to customer *)
$-[trans]_{pvt(c)} \cdot \{od\}_{pub(m)} \cdot epd,$ (* 3. from customer *)
$+[trans]_{pvt(c)} \cdot [trans]_{pvt(m)} \cdot epd,$ (* 4. to gateway *)
$-[result \cdot h(trans)]_{pvt(g)} \rangle\!\rangle$ (* 5. from gateway *)

Gate($c : Name_c \cup \{P\}$, $m : Name_m \cup \{P\}$, $g : Name_g$, $nc : Nonce$, $nm : Nonce$,
$price : Price$, $hod : Hash(Order)$, $pd : PayDesc$, $result : Result$) =
let $trans = c \cdot m \cdot g \cdot nc \cdot nm \cdot price \cdot hod \cdot h(pd)$ **in**
$\langle\!\langle -[trans]_{pvt(c)} \cdot [trans]_{pvt(m)} \cdot \{pd\}_{pub(g)}$ (* 1. from merchant *)
$+[result \cdot h(trans)]_{pvt(g)} \rangle\!\rangle$ (* 2. to merchant *)

Figure 1: Roles for Π_{SET}. Comments indicate step number and intended source or destination of message.

Let $\mathrm{Hist}(\Pi)$ denote the set of histories of a protocol Π.

A set S of nodes is *backwards-closed* with respect to a binary relation R iff, for all nodes n_1 and n_2, if $n_2 \in S$ and $n_1 \; R \; n_2$, then $n_1 \in S$.

Given a history h of a protocol Π, a set S of nodes of h that is backward-closed with respect to $\preceq_h$ can be regarded as a history, denoted $\mathrm{nodesToHist}_h^{\Pi}(S)$, in a natural way. Specifically, $\mathrm{nodesToHist}_h^{\Pi}(S)$ is $\langle tr_1, \overset{msg}{\rightarrow}_1, role_1 \rangle$, where $\mathcal{N}_{tr_1} = S$, $\overset{msg}{\rightarrow}_1 = \overset{msg}{\rightarrow} \cap (S \times S)$, $\mathrm{term}_{tr_1}(n) = \mathrm{term}_{tr}(n)$ for all $n \in S$, and $role_1(s) = role(s)$ for all $s \in \mathrm{dom}(tr_1)$.

2.3 Examples

Consider a payment protocol Π_{SET} based closely on [Bol97] and reminiscent of SET [SET97], including the use of a dual-signature technique, so that the customer produces only one digital signature. Let $Order \subset Text$ and $PayDesc \subset Text$ denote sets of order and payment descriptions, respectively. Let $Price \subset Text$ and $Result \subset Text$ denote sets of prices and results (*e.g.*, "approved"), respectively. We assume these subsets of $Text$ are disjoint. Let $Name_c$, $Name_m$, and $Name_g$ be disjoint subsets of $Name$ not containing P. For a set S of terms, let $Hash(S) = \{h(t) \mid t \in S\}$. The roles of protocol Π_{SET} appear in Figure 1, and $\Pi_{\mathrm{SET}}.Secret = \emptyset$, for reasons given below. We use **let** expressions to avoid repetition of large subterms. We allow $\mathrm{Cust}.m = P$ and $\mathrm{Gate}.m = P$ to model malicious merchants; similarly for malicious clients and gateways. There is no reason to allow the "me" variable of each role (namely, $\mathrm{Cust}.c$, $\mathrm{Mrch}.m$, and $\mathrm{Gate}.g$) to equal P, because P's actions are modeled by penetrator strands.

Use of $Hash(PayDesc)$ instead of the set of all hashes as the type for $\mathrm{Mrch}.hpd$ requires some justification, because a merchant cannot determine whether the hash received in hpd is the hash of a payment description or, say, a ciphertext. Attacks involving terms that are not of the expected type are called *type flaw attacks*.

Cust($od : Order$, $price : Price$, $\underline{salt_c} : Nonce$, $\underline{R_c} : Nonce$, $CAN : AcctNum_0$,
$ID_m : Name_m \cup \{P\}$, $TID_m : Nonce$, $nonce_m : Nonce$, $g : Name_g$, $YesNo : Result$) =
let $cid = h(R_c \cdot CAN)$
and $common = price \cdot ID_m \cdot TID_m \cdot nonce_m \cdot cid \cdot h(od \cdot salt_c)$
and $clear = ID_m \cdot TID_m \cdot nonce_m \cdot h(common)$
and $slip = price \cdot h(common) \cdot CAN \cdot R_C$ **in**

$\langle\!\langle +salt_c \cdot cid,$	$(*$ 1. to merchant $*)$
$\ -clear$	$(*$ 2. from merchant $*)$
$\ +\{slip\}_{pub(g)},$	$(*$ 3. to merchant $*)$
$\ -YesNo \cdot [h(YesNo \cdot h(common))]_{pvt(g)} \rangle\!\rangle$	$(*$ 4. from merchant $*)$

Mrch($od : Order$, $price : Price$, $salt_c : Nonce$, $cid : Hash(Nonce \times AcctNum)$, $ID_m : Name_m$,
$TID_m : Nonce$, $\underline{nonce_m} : Nonce$, $g : Name_g$, $YesNo : Result$, $eslip : Term$) =
let $common = price \cdot ID_m \cdot TID_m \cdot nonce_m \cdot cid \cdot h(od \cdot salt_c)$
and $clear = ID_m \cdot TID_m \cdot nonce_m \cdot h(common)$ **in**

$\langle\!\langle -salt_c \cdot cid,$	$(*$ 1. from customer $*)$
$\ +clear,$	$(*$ 2. to customer $*)$
$\ -eslip,$	$(*$ 3. from customer $*)$
$\ +clear \cdot h(od \cdot salt_c) \cdot eslip,$	$(*$ 4. to gateway $*)$
$\ -YesNo \cdot [h(YesNo \cdot h(common))]_{pvt(g)},$	$(*$ 5. from gateway $*)$
$\ +YesNo \cdot [h(YesNo \cdot h(common))]_{pvt(g)} \rangle\!\rangle$	$(*$ 6. to customer $*)$

Gate($price : Price$, $R_c : Nonce$, $CAN : AcctNum$, $ID_m : Name_m \cup \{P\}$,
$TID_m : Nonce$, $nonce_m : Nonce$, $g : Name_g$, $hodsalt : Hash(Order \times Nonce)$, $YesNo : Result$) =
let $cid = h(R_c \cdot CAN)$
and $common = price \cdot ID_m \cdot TID_m \cdot nonce_m \cdot cid \cdot hodsalt$
and $clear = ID_m \cdot TID_m \cdot nonce_m \cdot h(common)$
and $slip = price \cdot h(common) \cdot CAN \cdot R_C$ **in**

$\langle\!\langle -clear \cdot hodsalt \cdot \{slip\}_{pub(g)},$	$(*$ 1. from merchant $*)$
$\ +YesNo \cdot [h(YesNo \cdot h(common))]_{pvt(g)} \rangle\!\rangle$	$(*$ 2. to merchant $*)$

Figure 2: Roles for $\Pi_{1\mathrm{KP}}$.

Use of the types $Hash(PayDesc)$ and $Hash(Order)$ can be justified by results like those in [HLS00], which show that type flaw attacks can be prevented by using type tags in the protocol implementation. Extending their results to accommodate hashing and to accommodate the slightly larger class of agreement properties introduced below is fairly straightforward.

As another example, consider a version of the 1KP protocol [BGH+00] based closely on [CMJ98]. Following [CMJ98], we assume the customer account number (CAN) is secret and hence (for brevity) omit the PIN. We also omit the date field, since it does not affect the secrecy or agreement properties of $\Pi_{1\mathrm{KP}}$ given below, assuming nonces are uniquely-originated. Let $AcctNum \subset Text$ be a set of account numbers. To model dishonest customers (*i.e.*, customers that collude with the penetrator), we partition $AcctNum$ into two sets, $AcctNum_0$ and $AcctNum_1$, which contain account numbers of honest and dishonest customers, respectively. Let $Order$, $Result$, $Name_m$, and $Name_g$ be as above. We assume these subsets of $Text$ are disjoint. 1KP is designed for settings where the gateway has a private key with a well-known public key, but the customer and merchant do not. Consequently, 1KP provides few guarantees if the gateway is dishonest, so we do not include P in the types of Cust.g and Mrch.g. The roles of protocol $\Pi_{1\mathrm{KP}}$ appear in Figure 2, and $\Pi_{1\mathrm{KP}}.Secret = AcctNum_0$.

2.4 Derivability

Informally, a term t is derivable (by the pentrator) from a set S of nodes if the penetrator can compute t from $\text{term}(S)$ and Key_P. A formal definition follows.

For a nonce g that uniquely originates in a history h, let $\text{origin}_h(g)$ denote the node from which g originates in h.

For a set S of nodes in a history $h = \langle tr, \stackrel{msg}{\rightarrow}, role \rangle$ of a protocol Π, let $\text{uniqOrigReqrd}_h^\Pi(S)$ denote the set of nonces g such that there exists $\langle s, i \rangle \in S$ and $x \in \text{dom}(args(role(s), tr(s)))$ such that parameter x is uniquely originated and $args(role(s), tr(s))(x) = g$ and $\text{origin}_h(g) = \langle s, i \rangle$.

For a directed term t, the absolute value of t, denoted $\text{abs}(t)$, is t without its sign. For $T \subseteq Term$, $\text{abs}(T) = \{\text{abs}(t) \mid t \in T\}$, and the role Src_T is defined by $\text{Src}_T(x : T) = \langle\!\langle +x \rangle\!\rangle$. When the subscript on Src is clear from context, we elide it.

A term t is *derivable* (by the penetrator) from a set S of nodes of a history h of a protocol Π, denoted $S \vdash_h^\Pi t$, if there exists a history $h' = \langle tr', \stackrel{msg'}{\rightarrow}, role' \rangle$ of the protocol $\{\text{Src}_{\text{abs}(\text{term}_h(S))}\}$ such that: (1) arguments of strands for Message in h' are not in $\text{uniqOrigReqrd}_h^\Pi(S)$ (*i.e.*, for all $s \in \text{dom}(tr')$, if $role'(s) = M$ and $x \in \text{dom}(args(M, tr(s')))$, then $args(M, tr(s'))(x) \notin \text{uniqOrigReqrd}_h^\Pi(S)$); and (2) there exists a node $n \in \mathcal{N}_{tr'}$ with $\text{term}_{tr'}(n) = +t$. The derivability relation, like the penetrator's Key role, is implicitly parameterized by the set Key_P of keys initially known to the penetrator. This relation is equivalent to the derivability relation in [CJM98] and can be computed using the approach in [CJM98]. Similar relations or functions have been considered by other researchers.

2.5 Correctness Requirements

We consider the following kinds of correctness requirements. For a correctness requirement ϕ, we say that a protocol Π satisfies ϕ iff every history of Π satisfies ϕ.

Long-Term Secrecy. A history h of a protocol Π satisfies long-term secrecy iff, for every $t \in \Pi.Secret \cup (Key \setminus Key_P)$, $\mathcal{N}_h \not\vdash_h^\Pi t$.

Nonce Secrecy. Informally, nonce secrecy says: the values of specified nonce parameters are not revealed to the penetrator. A nonce secrecy requirement has the form "$r.x$ is secret unless $r.y \in S$", where $r \in \Pi$, x and y are parameters of r, and $S \subseteq Text$ (typically, $S \subseteq Name$). A history $h = \langle tr, \stackrel{msg}{\rightarrow}, role \rangle$ of a protocol Π satisfies that requirement iff, for every strand $s \in \text{dom}(tr)$, if $role(s) = r$ and $y \in \text{dom}(args(role(s), tr(s)))$ and $args(role(s), tr(s))(y) \notin S$, then $\mathcal{N}_{tr} \not\vdash_h^\Pi args(role(s), tr(s))(x)$.

Agreement. Informally, agreement says: if some strand executed a certain role to a certain point with certain arguments, then some strand must have executed a corresponding role to a corresponding point with corresponding arguments. An agreement requirement has the form "$\langle r_2, len_2 \rangle$ satisfying $x_2 \notin S_2$ is preceded by $\langle r_1, len_1 \rangle$ satisfying $t_1 = t_2$", where x_2 is a parameter of r_2, S_2 is a subset of $Text$, and t_1 and t_2 are terms containing parameters of r_1 and r_2, respectively, as free variables. A history $h = \langle tr, \stackrel{msg}{\rightarrow}, role \rangle$ of a protocol Π satisfies that agreement requirement iff, if h contains a strand s_2 such that $role(s_2) = r_2$, $\text{len}(tr(s_2)) \geq len_2$, and $args(r_2, tr(s_2))(x_2) \notin S_2$, then tr contains a strand s_1 for role r_1 such that $\text{len}(tr(s_1)) \geq len_1$ and t_1 instantiated with the arguments of s_1 equals t_2 instantiated with the arguments of s_2.

One of Bolignano's requirements for Π_{SET} is that the gateway has proof of transaction authorization by the merchant [Bol97, p. 12]. This can be expressed as an agreement requirement: $\langle \text{Gate}, 1 \rangle$ satisfying

Gate.$m \notin \{P\}$ is preceded by $\langle$Mrch, 4$\rangle$ satisfying

> **let** $trans_m$ = Mrch.c·Mrch.m·Mrch.nc·Mrch.nm·Mrch.$price$·h(Mrch.od)·Mrch.hpd
> **and** $trans_g$ = Gate.c·Gate.m·Gate.nc·Gate.nm·Gate.$price$·Gate.hod·h(Gate.pd) **in**
> $trans_m = trans_g \wedge$ Mrch.g = Gate.g

This requirement applies even if Gate.$c = P$, *i.e.*, even if the customer is dishonest.[3] SET is designed to provide secrecy for order and payment descriptions. Π_{SET} as defined above does not provide such secrecy, because, *e.g.*, a customer strand with Cust.$m = P$ can reveal an order description to the penetrator. This is why we take $\Pi_{\text{SET}}.Secret = \emptyset$. To express secrecy of order descriptions from gateways, we use a variant Π^o_{SET} in which merchants are assumed to be honest; specifically, Π^o_{SET} differs from Π_{SET} as follows: the type for Cust.m is $Name_m$, and $\Pi^o_{\text{SET}}.Secret = Order$. Dishonest gateways are modeled by penetrator strands (the types of Cust.g and Mrch.g contain P), so if order descriptions are not known to the penetrator, then they are not known to dishonest gateways, so they are not known to honest gateways. Similarly, to express secrecy of payment descriptions from merchants, we use a variant Π^p_{SET} in which gateways are assumed to be honest; specifically, Π^p_{SET} differs from Π_{SET} as follows: the type for Cust.g is $Name_g$, and $Secret = PayDesc$.

Using multiple versions of the protocol to express the requirements is slightly awkward but has the virtue of simplicity. One alternative is to annotate the protocol with predicates indicating the conditions under which the value of each parameter whose type intersects *Secret* may be revealed to the adversary; for example, for Cust.od, the predicate would be Cust.$m = P$. This alternative leads to stronger long-term secrecy requirements for some (contrived) protocols. We do not adopt it because it is more complicated, and the secrecy requirements are equivalent for reasonable protocols. Modifying our reductions to accommodate that approach is straightforward.

Bellare *et al.* specify agreement requirements for 1KP. For example, their requirement A1 (gateway has proof of transaction authorization by customer) is roughly: $\langle$Gate, 1$\rangle$ satisfying Gate.$CAN \notin AcctNum_1$ is preceded by $\langle$Cust, 3$\rangle$ satisfying

> **let** cmn_g = Gate.$price$·Gate.ID_m·Gate.TID_m·Gate.$nonce_m$·Gate.R_c·Gate.CAN·Gate.$hodsalt$
> **and** cmn_c = Cust.$price$·Cust.ID_m·Cust.TID_m·Cust.$nonce_m$·Cust.R_c·Cust.CAN·h(od, Cust.$salt_c$)
> **in** $cmn_g = cmn_c$

Their requirement M1 (merchant has proof of transaction authorization by gateway) can be expressed similarly. Long-term secrecy expresses secrecy of account numbers in $AcctNum_0$ from merchants (recall that Π_{1KP} models dishonest merchants by allowing Cust.$ID_m = P$ and Gate.$ID_m = P$). To express secrecy of order descriptions from gateways, we use a variant Π^o_{1KP} of Π_{1KP} in which merchants are assumed to be honest, but gateways may be dishonest; specifically, Π^o_{1KP} differs from Π_{1KP} as follows: the type for Cust.ID_m does not contain P, the types for Cust.g and Mrch.g contain P, and $Secret = Order$. The nonce secrecy requirement is: Cust.R_c is secret unless Cust.$g \in \{P\}$.

3 Support

Informally, a set S' of nodes of a history tr supports a set S of nodes of tr if $S' \supseteq S$ and S' contains all of the regular nodes on which regular nodes in S depend. A formal definition follows.

For $T \subseteq Term$, the set of nonces that occur in T is nonces(T) = $\{g \in Nonce \mid \exists t \in T : g$ occurs in $t\}$.

[3] Bolignano's version of the protocol omits g from *trans* and consequently violates the conjunct Mrch.g = Gate.g (in his presentation, this conjunct corresponds to $st'.mcht.gateway = G$ in the second filter function on p. 12).

Let $\mathcal{RN}_h^\Pi$ denote the set of regular nodes in history h of protocol Π.

A set S' of nodes is a *support* for a set S of nodes in a history h of a protocol Π if:

1. $\mathcal{N}_h \supseteq S' \supseteq S$.
2. S' is backwards-closed with respect to $\stackrel{lcl}{\rightarrow}$.
3. For all negative nodes n in S', $\text{preds}_h(n) \cap S' \cap \mathcal{RN}_h^\Pi \vdash_h^\Pi \text{term}_h(n)$.
4. For all $g \in \text{nonces}(\text{term}_h(S')) \cap (\text{uniqOrigReqrd}_h^\Pi(\mathcal{N}_h) \setminus \text{uniqOrigReqrd}_h^\Pi(S'))$, g occurs in the clear in $\text{term}_h(\text{origin}_h(g))$. (This condition ensures the compositionality property expressed in Lemma 2.)

If S' is a support for S, we say S' supports S. For a strand s, if S' supports $\text{nodes}(s)$, we say that S' supports s.

For example, consider the following history of a generic payment protocol. Suppose $s_{c,1}$, $s_{m,1}$, and $s_{g,1}$ are customer, merchant, and gateway strands, respectively, that interact without interference from the penetrator. Let g be a nonce that uniquely originates on $s_{m,1}$ and is revealed to the penetrator (*e.g.*, the value of Mrch.nm in Π_{SET}, or $nonce_m$ in Π_{1KP}). The penetrator then behaves as a merchant, interacting with a customer strand $s_{c,2}$ and a gateway strand $s_{g,2}$, except that the penetrator uses g instead of a fresh nonce. A support for $s_{c,2}$ or $s_{g,2}$ need not contain nodes on $s_{m,1}$ or $s_{c,1}$. In that sense, $s_{c,2}$ and $s_{g,2}$ do not depend on $s_{m,1}$, even though the chain of messages that conveys g means that there is causal dependence between those nodes in the classical sense of Lamport [Lam78]. Informally, that classical dependence can be ignored here because the penetrator could generate a nonce g' and replace g with g' in the terms of nodes on $s_{c,2}$ and $s_{g,2}$. The careful treatment of unique origination in the definition of derivability allows such inessential classical dependencies to be ignored. The following lemma says that a support can be transformed into a history by adding penetrator nodes, without adding or changing regular nodes.

For a set S of nodes, let $\text{strand}(S) = \{\text{strand}(n) \mid n \in S\}$. For a trace mapping tr, a strand $s \in \text{dom}(tr)$, and a set S of nodes of tr that is backwards-closed with respect to $\stackrel{lcl}{\rightarrow}$, S contains nodes on a prefix of $tr(s)$; let $\text{prefix}_{tr}(s, S)$ denote that prefix.

Lemma 1. Let Π be a protocol. If S' is a support for S in a history $h = \langle tr, \stackrel{msg}{\rightarrow}, role\rangle$ of Π, then there exists a history $h' = \langle tr', \stackrel{msg'}{\rightarrow}, role'\rangle$ of Π such that

$$\begin{array}{l}(\forall s \in \text{strand}(S') : s \in \text{dom}(tr') \ \wedge \ tr'(s) = \text{prefix}_{tr}(s, S') \ \wedge \ role'(s) = role(s)) \\ \wedge \ (\forall s \in \text{dom}(tr') \setminus \text{strand}(S') : role'(s) \in \Pi_P(Key_P)) \\ \wedge \ (\forall n_1, n_2 \in S' : n_1 \stackrel{msg'}{\rightarrow} n_2 \Rightarrow n_1 \stackrel{msg}{\rightarrow} n_2)\end{array} \tag{1}$$

Proof: A witness h' can be constructed as follows. Let S'_{neg} denote the set of negative nodes in S'. For each $n \in S'_{neg}$, let $h_n = \langle tr_n, \stackrel{msg}{\rightarrow}_n, role_n\rangle$ be a history that witnesses the truth of $\text{preds}_h(n) \cap S' \cap \mathcal{RN}_h^\Pi \vdash_h^\Pi \text{term}_{tr}(n)$. Let out_n be a node in $\mathcal{N}_{tr_n}$ such that $\text{term}_{tr_n}(out_n) = +\text{abs}(\text{term}_{tr}(n))$. For each strand s for Src in h_n, let $\text{witness}_n(s)$ be a node in $\text{preds}_h(n) \cap S' \cap \mathcal{RN}_h^\Pi$ such that $\text{abs}(\text{term}_{tr}(\text{witness}_n(s))) = args(\text{Src}, tr_n(s))(x)$; the definitions of support and derivability together imply that such a node exists. Rename strands, if necessary, so that $\text{dom}(tr_n) \cap \text{dom}(tr) = \emptyset$ and $\text{dom}(tr_{n_1}) \cap \text{dom}(tr_{n_2}) = \emptyset$ for $n_1 \neq n_2$. We define h' in two steps. The first step merges S' and all the h_n (for n in S'_{neg}), yielding $\langle tr_1, role_1, \stackrel{msg}{\rightarrow}_1\rangle$. The second step eliminates strands for Src, yielding h'.

$$\text{dom}(tr_1) \quad = \quad \text{strand}(S') \cup \bigcup_{n \in S'_{neg}} \text{dom}(tr_n)$$

$$
\begin{aligned}
tr_1(s) &= \begin{cases} \text{prefix}_{tr}(s, S') & \text{if } s \in \text{strand}(S') \\ tr_n(s) & \text{if } s \in \text{dom}(tr_n) \text{ for some } n \in S'_{neg} \end{cases} \\
role_1(s) &= \begin{cases} role(s) & \text{if } s \in \text{strand}(S') \\ role_n(s) & \text{if } s \in \text{dom}(tr_n) \text{ for some } n \in S'_{neg} \end{cases} \\
\overset{msg}{\rightarrow}_1 &= \overset{msg}{\rightarrow} \cap (S' \times S') \cup \bigcup_{n \in S'_{neg}} \overset{msg}{\rightarrow}_n \cup \{\langle out_n, n \rangle\} \\
\text{dom}(tr') &= \{s \in \text{dom}(tr_1) \mid role_1(s) \neq \text{Src}\} \\
tr'(s) &= tr_1(s) \\
role'(s) &= role_1(s) \\
\text{source}_n(s) &= \begin{cases} \text{witness}_n(s) & \text{if } \text{term}_{tr}(\text{witness}_n(s)) \text{ is positive} \\ out_{\text{witness}_n(s)} & \text{if } \text{term}_{tr}(\text{witness}_n(s)) \text{ is negative} \end{cases} \\
\overset{msg'}{\rightarrow} &= \overset{msg}{\rightarrow}_1 \cap (\mathcal{N}_{tr'} \times \mathcal{N}_{tr'}) \cup \bigcup_{n \in S'_{neg}} \{\langle \text{source}_n(s), n_1 \rangle \mid s \in \text{dom}(tr_1) \wedge role_1(s) = \text{Src} \wedge \langle s, 0 \rangle \overset{msg}{\rightarrow}_1 n_1\}
\end{aligned}
$$

$\langle tr', \overset{msg'}{\rightarrow}, role' \rangle$ being a history of Π follows from h and all the h_n being histories of the appropriate protocols and from the following observations. For acyclicity of $\overset{msg'}{\rightarrow}$, note that $\text{witness}_n(s) \overset{msg}{\rightarrow} n$. For the unique-origination condition, we need to show that for every regular strand $s \in \text{strand}(S')$, for every parameter $x \in \text{dom}(args(role'(s), tr'(s)))$, if x is uniquely-originated, then the nonce $g = args(role'(s), tr'(s))(x)$ uniquely originates from $\langle s, i \rangle$, where i is the index of the first term in r that contains x. Note that $role'(s) = role(s)$, $tr'(s) = \text{prefix}_{tr}(s, S')$, $g \in \text{uniqOrigReqrd}_h^{\Pi}(S')$, and $\langle s, i \rangle \in S'$. h is a history of Π, so g does not originate from any other node in S'. It remains to show that g does not originate from any penetrator node in tr'. By inspection of $\Pi_P(Key_P)$, a nonce that originates from a penetrator node must originate from a strand for Message. The definitions of support and derivability imply that tr' does not contain strands for Message from which nonces in $\text{uniqOrigReqrd}_h^{\Pi}(S')$ originate. ∎

Lemma 2. If S'_0 and S'_1 support S_0 and S_1, respectively, in a history $h = \langle tr, \overset{msg}{\rightarrow}, role \rangle$ of a protocol Π, then $S'_0 \cup S'_1$ supports $S_0 \cup S_1$ in history h of Π.

Proof: It is easy to show that the first, second, and fourth conditions in the definition of support are satisfied. For the third condition, consider $i \in \{0, 1\}$, and consider a negative node n in S'_i. Let $h' = \langle tr', \overset{msg'}{\rightarrow}, role' \rangle$ be a history that witnesses $\text{preds}_h(n) \cap S'_i \cap \mathcal{RN}_h^{\Pi} \vdash_h^{\Pi} \text{term}_h(n)$. Suppose $\text{term}_h(n)$ does not contain a nonce in $\text{uniqOrigReqrd}_h^{\Pi}(S'_{(i+1)\%2}) \setminus \text{uniqOrigReqrd}_h^{\Pi}(S'_i)$; then h' also witnesses $\text{preds}_h(n) \cap (S'_0 \cup S'_1) \cap \mathcal{RN}_h^{\Pi} \vdash_h^{\Pi} \text{term}_h(n)$. Suppose $\text{term}_h(n)$ contains a nonce g in $\text{uniqOrigReqrd}_h^{\Pi}(S'_{(i+1)\%2}) \setminus \text{uniqOrigReqrd}_h^{\Pi}(S'_i)$; then h' might not be a witness for $\text{preds}_h(n) \cap (S'_0 \cup S'_1) \cap \mathcal{RN}_h^{\Pi} \vdash_h^{\Pi} \text{term}_h(n)$, because h' might contain strands for Message with argument g, violating the first condition in the definition of derivable. Note that $\text{origin}_h(g) \in S'_{(i+1)\%2}$, and $\text{origin}_h(g) \preceq_h n$ (because g originates from $\text{origin}_h(g)$ in h, and $\text{term}_h(n)$ contains g), and $\text{origin}_h(g) \in \mathcal{RN}_h^{\Pi}$ (because penetrator roles do not have uniquely-originated parameters). Thus, we can construct a history h'' that witnesses $\text{preds}_h(n) \cap (S'_0 \cup S'_1) \cap \mathcal{RN}_h^{\Pi} \vdash_h^{\Pi} \text{term}_h(n)$ by starting with h' and, for each nonce g in $\text{uniqOrigReqrd}_h^{\Pi}(S'_{(i+1)\%2}) \setminus \text{uniqOrigReqrd}_h^{\Pi}(S'_i)$,

1. Add a strand s_{Src} for Src with argument $\text{term}_h(\text{origin}_h(g))$. Thus, $\text{term}_{h''}(\langle s_{\text{Src}}, 0 \rangle) = \text{term}_h(\text{origin}_h(g))$.
2. Add strands for Separation, if necessary, to select g from $\text{term}_{h''}(\langle s_{\text{Src}}, 0 \rangle)$, so h'' contains a node n_g with $\text{term}(n_g) = +g$. This is possible because the fourth condition in the definition of support implies that g occurs in the clear in $\text{term}_h(\text{origin}_h(g))$.

3. For each strand s_M for Message with argument g, delete s_M from h'' and, for each negative node n_{neg} such that $\langle s_M, 0\rangle \overset{msg'}{\rightarrow} n_{neg}$, let $n_g \overset{msg''}{\rightarrow} n_{neg}$. This does not create cycles in $\preceq_{h''}$; the main point is that s_{Src} contains no negative nodes, so $\langle s_{\mathrm{Src}}, 0\rangle$ is $\preceq_{h''}$-minimal. ∎

3.1 Bounded Support Restriction

A *strand count* for a protocol Π is a function from the roles of Π to the natural numbers. A set S of nodes *has strand count* f iff, for each role r, S contains nodes from exactly $f(r)$ strands for r. If $\mathcal{N}_h$ has strand count f, then we say that history h has strand count f. Let $f_1(r) = 1$ for every role r. We define a partial ordering $\preceq_{SC}$ on strand counts for a protocol; $\preceq_{SC}$ is simply the pointwise extension of the standard ordering on natural numbers.

A history h satisfies the *bounded support restriction*, abbreviated BSR, iff for each regular strand s in h, there exists a support for s in h with strand count at most f_1. A protocol satisfies BSR iff all of its histories do.

Π_{SET} and Π_{1KP} satisfy BSR. We proved these results manually; the proofs are similar to the proof in [Sto99] for Lowe's corrected version of the Needham-Schroeder public-key authentication protocol. Theorem 2 in Section 5 shows that in principle, these results can be obtained automatically by state-space exploration of histories with bounded strand counts; an algorithm like the one in [Sto99] can be used to compute a (small) support for a given set of nodes. The current bounds probably need to be decreased somewhat before this is feasible, *e.g.*, by finding a tighter bound on the dependence width (see Section 4).

4 Dependence Width

Informally, the dependence width of a negative term $r(i)$ in a role r of a protocol Π, denoted $\mathrm{DW}(\langle r, i\rangle, \Pi)$, is the maximum number of "additional" positive regular nodes needed in any history h of Π to provide the penetrator with enough knowledge to produce the term received by any node $\langle s, i\rangle$ of h such that $role(s) = r$. "Additional" here means "beyond those needed for the penetrator to produce negative terms that occur earlier in the same strand". The dependence width of a protocol Π, denoted $\mathrm{DW}(\Pi)$, is the maximum over all negative terms $r(i)$ in roles r in Π of $\mathrm{DW}(\langle r, i\rangle, \Pi)$. The concept of dependence width is used in the proof of Theorem 2 in Section 5 to bound the number of strands involved in a violation of BSR.

Let n be a negative node of a history h of a protocol Π, and let t be a subterm of $\mathrm{term}_h(n)$. A *revealing set* for t at n in h is a set S of positive regular nodes of tr such that $S \subseteq \mathrm{preds}_h(n)$ and $S \vdash_h^{\Pi} t$.[4]

For a set S of numbers, let $\min(S)$ and $\max(S)$ denote the minimum and maximum element of S, respectively. We define $\min(\emptyset) = 0$ and $\max(\emptyset) = 0$.

The *revealing set min-size* of t at $\langle s, i\rangle$ in h is

$$\mathrm{rvlSetMinSz}(t, \langle s, i\rangle, h) = \min(\{\mathrm{size}(R \setminus \mathrm{nodes}_h(s)) \mid R \text{ is a revealing set for } t \text{ at } \langle s, i\rangle \text{ in } h\}) \qquad (2)$$

Nodes in R that are on the same strand as n are not counted in the revealing set min-size (and hence not in the dependence width), because in the proof of Theorem 2—specifically, in equation (5)—those nodes appear in $\mathrm{support}_{h_0}^{\Pi}(s_0)$ and hence are excluded from the index set of the rightmost union, and the dependence width is designed to bound the size of that index set.

Note that, if there are no revealing sets for t at n in h (*i.e.*, t is not known to the penetrator at that point), then $\mathrm{rvlSetMinSz}(t, n, h) = 0$.

[4] Roughly speaking, the main difference between a "revealing set for term(n) at n" and a "support for $\{n\}$" is that the former considers only one step of dependency, while the latter implicitly considers a transitive closure of dependencies.

Let r be a role in a protocol Π, and let i be the index of a negative term in r. The *dependence width* of $\langle r, i\rangle$ in Π is

$$\mathrm{DW}(\langle r, i\rangle, \Pi) = \max(\{\mathrm{rvlSetMinSz}(\mathrm{term}_{tr}(\langle s, i\rangle), \langle s, i\rangle, \langle tr, \overset{msg}{\rightarrow}, role\rangle) \mid \langle tr, \overset{msg}{\rightarrow}, role\rangle \in \mathrm{Hist}(\Pi) \wedge \langle s, i\rangle \in \mathcal{N}_{tr} \wedge role(s) = r\}) \quad (3)$$

The *dependence width* of a protocol Π is

$$\mathrm{DW}(\Pi) = \max(\{\mathrm{DW}(\langle r, i\rangle, \Pi) \mid r \in \Pi \wedge r(i) \text{ is a negative term}\}) \quad (4)$$

The proof of Theorem 2, and therefore also the proof of Theorem 3, rely on an upper bound on the dependence width of the protocol. If the protocol might send terms of the forms $\{g\}_{k_1}$, $\{k_1\}_{k_2}$, $\{k_2\}_{k_3}$, ..., $\{k_{i-1}\}_{k_i}$, k_i, then $i+1$ terms are needed to reveal g to the penetrator. Our long-term secrecy requirement prohibits such behavior. *Secrecy-limited dependence width*, abbreviated SL dependence width and denoted $\mathrm{DW}_{\mathrm{SL}}$, is defined in the same way as dependence width, except that the maximum over histories is restricted to histories satisfying long-term secrecy.

Let Π be a protocol, and let t be a term, possibly containing parameters. $\mathrm{nSecret}_0(t, \Pi)$ is a bound on the number of subterms of t that are not known to the penetrator, ignoring keys and values of parameters; formally, $\mathrm{nSecret}_0(t, \Pi) = N_c + N_h + N_{prim}$, where N_c is the number of subterms of t whose outermost operator is *encr*, ignoring those whose second argument is always in Key_P (based on parameter types), N_h is the number of subterms of t with outermost operator h, and N_{prim} is the number of elements of $Nonce \cup \Pi.Secret$ that occur in t. In computing N_c and N_h, identical subterms are counted only once. For a parameter $r.x$ of a role r of Π, $\mathrm{nSecret}(r.x, \Pi) = \max(\{\mathrm{nSecret}_0(t, \Pi) \mid t \text{ is in the type of } r.x\})$. Let $\mathrm{nSecret}(\langle r, i\rangle, \Pi) = \mathrm{nSecret}_0(r(i), \Pi) + \sum_{x \in \mathrm{params}(r(i))} \mathrm{nSecret}(r.x, \Pi)$, where $\mathrm{params}(t)$ is the set of parameters that occur in t.

Theorem 1. Let $r(i)$ be a negative term in a role r of a protocol Π. $\mathrm{DW}_{\mathrm{SL}}(\langle r, i\rangle, \Pi) \leq \mathrm{nSecret}(\langle r, i\rangle, \Pi)$.

Proof: Consider a strand s for r in a history h for Π. We consider each subterm t_1 of $\mathrm{term}_h(\langle s, i\rangle)$ and show that each hash, ciphertext, and element of $\mathrm{uniqOrigReqrd}_h^{\Pi}(\mathcal{N}_h) \cup \Pi.Secret$ that occurs in $\mathrm{term}_h(\langle s, i\rangle)$ contributes at most 1 to $\mathrm{DW}_{\mathrm{SL}}(\langle r, i\rangle, \Pi)$. The number of such subterms is bounded by $\mathrm{nSecret}(\langle r, i\rangle, \Pi)$. Other subterms contribute nothing. The definition of dependence width implies that terms not derivable by the penetrator contribute nothing to the dependence width (because such terms have no revealing sets), so in computing the bound, we conservatively assume all subterms are derivable by the penetrator. Consider cases based on the type of t_1.

case 1: $t_1 \in Key$. Long-term secrecy implies that no keys are revealed, so keys contribute nothing to $\mathrm{DW}_{\mathrm{SL}}(\langle r, i\rangle, \Pi)$.

case 2: $t_1 \in \mathrm{uniqOrigReqrd}_h^{\Pi}(\mathcal{N}_h) \cup \Pi.Secret$. The definition of history implies that t_1 originates from a regular node in h and (according to the conservative assumption discussed above) is derivable by the penetrator (using strands for Separation and Decryption), so there is a positive regular node n such that t_1 occurs in $\mathrm{term}_h(n)$ either in the clear or encrypted only with keys known to the penetrator. Long-term secrecy implies that those keys (if any) are in Key_P. Thus, t_1 is derivable from $\{n\}$, so t_1 contributes at most 1 to $\mathrm{DW}_{\mathrm{SL}}(\langle r, i\rangle, \Pi)$.

case 3: $t_1 \in Text \setminus (\mathrm{uniqOrigReqrd}_h^{\Pi}(\mathcal{N}_h) \cup \Pi.Secret)$. t_1 is directly available to the penetrator through the Message role, so t_1 contributes nothing to $\mathrm{DW}_{\mathrm{SL}}(\langle r, i\rangle, \Pi)$.

case 4: t_1 is a pair. Revealing a pair is equivalent to revealing its two components, so proper subterms of t_1 contribute to $\mathrm{DW}_{\mathrm{SL}}(\langle r, i\rangle, \Pi)$, but t_1 itself does not.

case 5: t_1 is a ciphertext or hash, and t_1 originates from a penetrator node in $\text{preds}_h(\langle s, i\rangle)$. The penetrator performs the encryption or hashing to construct its copy of t_1, so proper subterms of t_1 contribute to $\text{DW}_{\text{SL}}(\langle r, i\rangle, \Pi)$, but t_1 itself does not.

case 6: t_1 is a ciphertext or hash, and t_1 does not originate from a penetrator node in $\text{preds}_h(\langle s, i\rangle)$. Then t_1 originates from a regular node, and the argument is the same as in case 2. Note that it is not necessary for proper subterms of t_1 to contribute to $\text{DW}_{\text{SL}}(\langle r, i\rangle, \Pi)$. Our bound on $\text{DW}_{\text{SL}}(\langle r, i\rangle, \Pi)$ might be loose because it does not attempt to exploit this observation; exploiting it is left for future work.

Now we justify ignoring, in the definition of N_c in nSecret_0, occurrences of *encr* whose second argument is always in Key_P. Let $\{t'\}_k$ be such a ciphertext.

case 1: $\emptyset \vdash_h^\Pi t'$; in other words, t' contains no secrets. Then $\emptyset \vdash_h^\Pi \{t'\}_k$, so $\{t'\}_k$ contributes nothing to $\text{DW}_{\text{SL}}(\langle r, i\rangle, \Pi)$.

case 2: $\emptyset \not\vdash_h^\Pi t'$; in other words, t' contains one or more secrets. Thus, subterms of t' contribute at least 1 to our bound on $\text{DW}_{\text{SL}}(\langle r, i\rangle, \Pi)$.

> **case** 2.1: $\text{preds}_h(\langle s, i\rangle) \vdash_h^\Pi t'$. The penetrator can perform the encryption to construct its copy of $\{t'\}_k$, so proper subterms of $\{t'\}_k$ contribute to $\text{DW}_{\text{SL}}(\langle r, i\rangle, \Pi)$, but $\{t'\}_k$ itself does not, so ignoring $\{t'\}_k$ in N_c is safe.
>
> **case** 2.2: $\text{preds}_h(\langle s, i\rangle) \not\vdash_h^\Pi t'$. The ciphertext $\{t'\}_k$ must originate from a regular node and be revealed to the penetrator. The ciphertext actually contributes 1 to $\text{DW}_{\text{SL}}(\langle r, i\rangle, \Pi)$ (*cf.* case 6 above), and its subterms actually contribute nothing. Our bound counts 0 from the ciphertext but counts at least 1 from subterms of t'. Thus, although the bookkeeping might seem skewed, the sum of the contributions is sufficient. ∎

We simplify Π_{SET} and Π_{1KP} as follows. Parameters *epd* and *eslip* are used to forward messages in a trivial way (specifically, all occurrences of these parameters are unencrypted), and TID_m is redundant because it always appears together with $nonce_m$. Thus, eliminating these parameters has no impact on correctness. Let Π'_{SET} and Π'_{1KP} refer to versions of the protocols in which these parameters have been eliminated. Theorem 1 implies $\text{DW}_{\text{SL}}(\Pi'_{\text{SET}}) \leq 6$ and $\text{DW}_{\text{SL}}(\Pi'_{\text{1KP}}) \leq 7$. In both protocols, the first term of Gate has the largest dependence width.

The bound on DW_{SL} provided by Theorem 1 can sometimes be decreased by replacing a negative term of the form $-t_1 \cdot t_2$ in a role with the sequence of terms $-t_1, -t_2$. For example, let Π''_{SET} denote the protocol obtained from Π'_{SET} by splitting the first term of Gate into a sequence of three terms. Theorem 1 implies $\text{DW}_{\text{SL}}(\Pi''_{\text{SET}}) \leq 5$. This transformation preserves all correctness requirements, provided the lengths in agreement requirements are adjusted appropriately. It can introduce violations of BSR, but this does not happen with Π'_{SET}.

5 Reduction for BSR and Long-Term Secrecy

The following lemma says, roughly, that constructing a history h' from a support S' of a set S of nodes of a history h does not create new supports for S.

Lemma 3. Suppose S_0 supports S in a history h of a protocol Π. Let h' be a history of Π whose existence is implied by Lemma 1 applied to S_0. Suppose S_1 supports S in history h' of Π. Then $S_1 \cap \mathcal{RN}_h^\Pi$ supports S in history h of Π.

Proof: The proof is similar to that of Lemma 3 in [Sto99]. ∎

For a protocol Π, define a strand count $\beta(\Pi)$ by $\beta(\Pi)(r) = \mathrm{DW}_{\mathrm{SL}}(\Pi) + 1$.

Theorem 2. A protocol Π satisfies BSR and long-term secrecy iff all histories of Π with strand count $\beta(\Pi)$ do.

Proof: The forward direction ($\Rightarrow$) of the "iff" is easy. For the reverse direction ($\Leftarrow$), we prove the contrapositive, *i.e.*, we suppose there exists a history h of Π that violates BSR or long-term secrecy, and we construct a history of Π with strand count at most $\beta(\Pi)$ that violates the same property.

BSR and long-term secrecy are safety properties [AS85] satisfied by histories with zero nodes, and $\preceq_h$ is well-founded, so there exists a $\preceq_h$-minimal node n_0 such that

1. $\mathrm{nodesToHist}_h^{\Pi}(\mathrm{preds}_h(n_0))$ satisfies BSR and long-term secrecy.
2. $\mathrm{nodesToHist}_h^{\Pi}(\mathrm{preds}_h(n_0)) \cup \{n_0\}$ violates BSR or long-term secrecy.

Let $h_0 = \mathrm{nodesToHist}_h^{\Pi}(\mathrm{preds}_h(n_0))$. Let $s_0 = \mathrm{strand}(n_0)$ and $i_0 = \mathrm{index}(n_0)$. Note that in h_0, s_0 does not include n_0. For a strand s in a history h' that satisfies BSR, let $\mathrm{support}_{h'}(s)$ denote a support for s in h' with strand count at most f_1 (recall that f_1 is defined in Section 3.1). The definitions of BSR and long-term secrecy imply n_0 is a regular node. Consider cases based on the sign of n_0.

case: n_0 is a negative node. n_0 cannot cause a violation of secrecy, so it causes a violation of BSR. Suppose $i_0 > 0$. n_0 directly depends on $\langle s_0, i_0 - 1\rangle$ and on a revealing set R for $\mathrm{term}(n_0)$ at n_0 in h; more precisely, for all S', if S' supports $\{\langle s_0, i_0 - 1\rangle\} \cup R$ in h, then $S' \cup \{n_0\}$ supports $\{n_0\}$ in h. h_0 satisfies long-term secrecy, so Theorem 1 implies $\mathrm{size}(R \setminus \mathrm{nodes}_{h_0}(s_0)) \leq \mathrm{DW}_{\mathrm{SL}}(\Pi)$. Let

$$S_1 = \{n_0\} \cup \mathrm{support}_{h_0}(s_0) \cup \bigcup_{n \in R \setminus \mathrm{nodes}_{tr_0}(s_0)} \mathrm{support}_{h_0}(\mathrm{strand}(n)). \tag{5}$$

h_0 satisfies BSR, so each of the supports in (5) has strand count at most f_1, so S_1 has strand count at most $\beta(\Pi)$ (note that n_0 is on s_0, so $\{n_0\} \cup \mathrm{support}_{h_0}(s_0)$ contributes at most f_1 to the strand count of S_1).

Lemma 2 implies that $S_1 \setminus \{n_0\}$ supports $\{\langle s_0, i_0 - 1\rangle\} \cup R$ in h; thus, S_1 supports $\{n_0\}$ in h. Lemma 1 implies that S_1 can be transformed into a history h_1 of Π by adding penetrator nodes. Adding penetrator nodes does not affect the strand count, so h_1 has strand count at most $\beta(\Pi)$. We show by contradiction that n_0 also causes a violation of BSR in h_1. Suppose n_0 does not cause such a violation. Then there exists a support S' for $\{n_0\}$ in h_1 with strand count at most f_1. Lemma 3 implies that $S' \cap \mathcal{RN}_{h_1}^{\Pi}$ is a support for $\{n_0\}$ in h with strand count at most f_1, a contradiction.

Suppose $i_0 = 0$. The proof is similar to the case $i_0 > 0$, except n_0 does not depend on the non-existent node $\langle s_0, i_0 - 1\rangle$, so we omit $\mathrm{support}_{h_0}(s_0)$ from the definition of S_1, and Lemma Lemma 2 implies that $S_1 \setminus \{n_0\}$ supports R in h.

case: n_0 is a positive node. n_0 cannot cause a violation of BSR, so it causes a violation of long-term secrecy. $\mathrm{preds}_h(n_0)$ satisfies long-term secrecy, so there is some $t \in \Pi.\mathit{Secret} \cup (\mathit{Key} \setminus \mathit{Key}_P)$ such that t appears in $\mathrm{term}_h(n_0)$ either in the clear or encrypted only with keys in Key_P. Suppose $i_0 > 0$. Let $S_0 = \mathrm{support}_{h_0}(s_0)$ and $S_1 = \{n_0\} \cup S_0$. h_0 satisfies BSR, so S_0 and S_1 have strand count at most f_1 (note that n_0 is on s_0, and $s_0 \in \mathrm{strand}(S_0)$, so n_0 does not increase the strand count of S_1). S_1 can be transformed into a history h_1 by adding penetrator nodes; this follows from Lemma 1 and the observation that n_0 is positive and is an immediate successor of the last node on s_0 in h_0. It is easy to show that adding penetrator nodes does not change the strand count or destroy the violation of long-term secrecy. Thus, h_1 is a history of Π with strand count at most $\beta(\Pi)$ that violates long-term secrecy. Suppose $i_0 = 0$. Then $\mathrm{preds}_h(n_0) = \emptyset$, and the history containing only node n_0 has strand count at most f_1 and violates long-term secrecy. ∎

6 Reduction for Nonce Secrecy and Agreement

Define a strand count f_2 by: $f_2(r) = 2$ for every role r.

Theorem 3. Let ϕ be a nonce secrecy or agreement requirement. Suppose all histories of a protocol Π with strand count $\beta(\Pi)$ satisfy BSR and long-term secrecy. Π satisfies ϕ iff all histories of Π with strand count f_2 do.

Proof: The forward direction ($\Rightarrow$) of the "iff" is easy. For the reverse direction ($\Leftarrow$), we prove the contrapositive, *i.e.*, we suppose there exists a history $h = \langle tr, \stackrel{msg}{\rightarrow}, role \rangle$ of Π that violates ϕ, and we construct a history of Π with strand count at most f_2 that violates ϕ. Nonce secrecy and agreement requirements are safety properties [AS85] satisfied by histories with zero nodes, and $\preceq_h$ is well-founded, so there exists a $\preceq_h$-minimal node n_0 such that

1. $\text{nodesToHist}_h^{\Pi}(\text{preds}_h(n_0))$ satisfies ϕ.
2. $\text{nodesToHist}_h^{\Pi}(\text{preds}_h(n_0)) \cup \{n_0\}$ violates ϕ.

Let $s_0 = \text{strand}(n_0)$.

By hypothesis, all histories of Π with strand count $\beta(\Pi)$ satisfy BSR and long-term secrecy, so Theorem 2 implies that Π satisfies BSR. For $s \in \text{dom}(h)$, let $\text{support}_h(s)$ denote a support for s with strand count at most f_1.

Suppose ϕ is a nonce secrecy requirement. ϕ has the form "$r.x$ is secret unless $r.y \in S$". n_0 is a positive regular node, and there is a regular strand s_g such that $\mathit{args}(\mathit{role}(s_g), tr(s_g))(y) \notin S$ and $\text{preds}_h(n_0) \not\vdash_h^{\Pi} g$ and $\text{preds}_h(n_0) \cup \{n_0\} \vdash_h^{\Pi} g$, where $g = \mathit{args}(\mathit{role}(s), tr(s))(x)$. By the same reasoning as in case 2 of the proof of Theorem 1, this implies that $\{n_0\} \vdash_h^{\Pi} g$. Let $S_1 = \text{support}_h(s_0) \cup \text{support}_h(s_g)$. Lemma 2 implies that S_1 is a support for $\text{nodes}_h(s_0) \cup \text{nodes}_h(s_g)$. Lemma 1 implies that S_1 can be transformed into a history h_1 by adding penetrator nodes. Note that S_1 and h_1 have strand count at most f_2. It is easy to see that n_0 causes a violation of nonce secrecy in h_1.

Suppose ϕ is an agreement requirement. ϕ has the form: "$\langle r_2, len_2 \rangle$ satisfying $x_2 \notin S_2$ is preceded by $\langle r_1, len_1 \rangle$ satisfying $t_1 = t_2$". n_0 causes a violation of ϕ, so s_0 is a strand for r_2 and $\mathit{args}(r_2, tr(s_2))(x_2) \notin S_2$ and $\text{index}(n_0) = len_2$. Lemma 1 implies that $\text{support}_h(s_0)$ can be transformed into a history h_0 of Π with strand count at most f_1. Note that $n_0 \in \mathcal{N}_{h_0}$. Removing nodes in $\mathcal{N}_h \setminus \mathcal{N}_{h_0}$ and adding penetrator nodes preserve the lack of a node $\langle s_1, len_1 \rangle$ such that $\mathit{role}(s_1) = r_1$ and such that t_1 instantiated with the arguments of s_1 equals t_2 instantiated with the arguments of s_0. Thus, h_0 violates ϕ. ∎

References

[AG99] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*, 143:1–70, 1999.

[AJ98] Parosh Aziz Abdulla and Bengt Jonsson. Verifying networks of timed processes. In *Proc. 4th Intl. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*. Springer-Verlag, 1998.

[AS85] Bowen Alpern and Fred B. Schneider. Defining liveness. *Information Processing Letters*, 24(4):181–185, October 1985.

[BGH+00] M. Bellare, J. A. Garay, R. Hauser, A. Herzberg, H. Krawczyk, M. Steiner, G. Tsudik, E. Van Herreweghen, and M. Waidner. Design, implementation and deployment of a secure

account-based electronic payment system. *IEEE Journal on Selected Areas in Communications*, 18(4):611–627, 2000.

[Bol97] Dominique Bolignano. Towards the formal verification of electronic commerce protocols. In *Proc. 10th IEEE Computer Security Foundations Workshop (CSFW)*. IEEE Computer Society Press, 1997.

[Bol98] Dominique Bolignano. Integrating proof-based and model-checking techniques for the formal verification of cryptographic protocols. In Alan J. Hu and Moshe Y. Vardi, editors, *Proc. Tenth Int'l. Conference on Computer-Aided Verification (CAV)*, volume 1427 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.

[BSBL00] Kai Baukus, Karsten Stahl, Saddek Bensalem, and Yassine Lakhnech. Abstracting ws1s systems to verify parameterized networks. In *Proc. 6th Intl. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 188–203. Springer-Verlag, 2000.

[CGJ95] Edmund M. Clarke, Orna Grumberg, and Somesh Jha. Verifying parameterized networks using abstractions and regular languages. In *Proc. Sixth Int'l. Conference on Concurrency Theory (CONCUR)*, 1995.

[CJM98] E.M. Clarke, S. Jha, and W. Marrero. Using state space exploration and a natural deduction style message derivation engine to verify security protocols. In *Proc. IFIP Working Conference on Programming Concepts and Methods (PROCOMET)*, June 1998.

[CMJ98] Edmund Clarke, Will Marrero, and Somesh Jha. A machine checkable logic of knowledge for specifying security properties of electronic commerce protocols. In *Proc. IFIP Working Conference on Programming Concepts and Methods (PROCOMET)*, June 1998.

[DK97] Z. Dang and R. A. Kemmerer. Using the ASTRAL model checker for cryptographic protocol analysis. In *Proc. DIMACS Workshop on Design and Formal Verification of Security Protocols*, September 1997.

[DNL99] Ben Donovan, Paul Norris, and Gavin Lowe. Analyzing a library of security protocols using Casper and FDR. In *Proc. 1999 Workshop on Formal Methods and Security Protocols*, July 1999. Available via http://cm.bell-labs.com/cm/cs/who/nch/fmsp99/.

[DY83] Danny Dolev and Andrew C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, IT-29(2):198–208, March 1983.

[EN96] E. Allen Emerson and Kedar S. Namjoshi. Automated verification of parameterized synchronous systems. In *Proc. 8th Int'l. Conference on Computer-Aided Verification (CAV)*, 1996.

[HLS00] James Heather, Gavin Lowe, and Steve Schneider. How to prevent type flaw attacks on security protocols. In *Proc. 13th IEEE Computer Security Foundations Workshop (CSFW)*. IEEE Computer Society Press, 2000.

[HTWW96] Nevin Heintze, J. D. Tygar, Jeannette Wing, and Hao-Chi Wong. Model checking electronic commerce protocols. In *Proc. USENIX 1996 Workshop on Electronic Commerce*, November 1996.

[JN00] Bengt Jonsson and Marcus Nilsson. Transitive closures of regular relations for verifying infinite state systems. In *Proc. 6th Intl. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 220–234. Springer-Verlag, 2000.

[KM95] R. P. Kurshan and K. L. McMillan. A structural induction theorem for processes. *Information and Computation*, 117(1):1–11, 1995.

[Lam78] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–564, 1978.

[Low99] Gavin Lowe. Towards a completeness result for model checking of security protocols. *The Journal of Computer Security*, 7(2/3):89–146, 1999.

[LR97] Gavin Lowe and Bill Roscoe. Using CSP to detect errors in the TMN protocol. *IEEE Transactions on Software Engineering*, 23(10):659–669, 1997.

[MCF87] Jonathan K. Millen, Sidney C. Clark, and Sheryl B. Freedman. The Interrogator: protocol security analysis. *IEEE Transactions on Software Engineering*, SE-13(2):274–288, February 1987.

[MCJ97] Will Marrero, Edmund Clarke, and Somesh Jha. A model checker for authentication protocols. In *Proc. DIMACS Workshop on Design and Formal Verification of Security Protocols*, 1997.

[MM99] Davide Marchignoli and Fabio Martinelli. Automatic verification of cryptographic protocols through compositional analysis techniques. In W. Rance Cleaveland, editor, *Proc. 5th Intl. Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 1579 of *Lecture Notes in Computer Science*, pages 148–162. Springer-Verlag, 1999.

[MMS97] John C. Mitchell, Mark Mitchell, and Ulrich Stern. Automated analysis of cryptographic protocols using Murϕ. In *Proc. 18th IEEE Symposium on Research in Security and Privacy*, pages 141–153. IEEE Computer Society Press, 1997.

[MSS98] John C. Mitchell, Vitaly Shmatikov, and Ulrich Stern. Finite-state analysis of SSL 3.0. In *Seventh USENIX Security Symposium*, pages 201–216, 1998.

[MvOV97] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[RB99] A. W. Roscoe and P. J. Broadfoot. Proving security protocols with model checkers by data independence techniques. *The Journal of Computer Security*, 7(2/3), 1999.

[Ros95] A. W. Roscoe. Modelling and verifying key exchange protocols using CSP and FDR. In *Proc. 8th IEEE Computer Security Foundations Workshop (CSFW)*, pages 98–107. IEEE Computer Society Press, 1995.

[SET97] SET: Secure Electronic Transaction Specification, version 1.0, May 1997. Available from www.setco.org.

[Sto99] Scott D. Stoller. A bound on attacks on authentication protocols. Technical Report 526, Computer Science Dept., Indiana University, July 1999. Submitted for journal publication.

[THG98] F. Javier Thayer Fábrega, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Why is a security protocol correct? In *Proc. 18th IEEE Symposium on Research in Security and Privacy*. IEEE Computer Society Press, 1998.