Protocol-Independent Secrecy

Jon Millen and Harald Rueß

SRI International
Menlo Park, CA 94025, USA
{millen,ruess}@csl.sri.com

Abstract

Inductive proofs of secrecy invariants for cryptographic protocols can be facilitated by separating the protocol-dependent part from the protocol-independent part. Our secrecy theorem encapsulates the use of induction so that the discharge of protocol-specific proof obligations is reduced to first-order reasoning. Also, the verification conditions are modularly associated with the protocol messages. Secrecy proofs for Otway-Rees and the corrected Needham-Schroeder protocol are given.

1 Introduction

Cryptographic protocols are used to achieve goals like authentication and key distribution. Security goals, as opposed to reliability goals, are typically expressed as safety properties, modeled as state invariants. In particular, secrecy or confidentiality is expressed by requiring that certain data items conveyed by the protocol are never visible to an unauthorized party. In this paper we concentrate on proving secrecy invariants. Secrecy has been shown to be undecidable even under very weak assumptions on the protocol [2], and secrecy proofs for individual protocols are often difficult. Our objective is to make secrecy proofs more practical, by isolating the protocol-specific parts of the proof from the universal parts that can be done ahead of time.

Our starting point is the inductive approach developed by Paulson [9]. In this model, a protocol is a rule for adding message events to a trace of prior events. A trace may involve many interleaved protocol runs. For purposes of analysis it is assumed that protocol messages sent over a network are also accessible to a hostile "spy" who is able to read, alter, and forge messages. Paulson uses a theorem prover to partially automate proofs by developing specialized strategies.

The main contribution of this paper is a theorem that reduces secrecy proofs for protocols to first-order reasoning; in particular, discharging these proof obligations does not require any inductions. The trick is to confine the inductions to general, protocol-independent lemmas, so that the protocol-specific part of the proof is minimized. The protocol-specific proof is is divided into cases associated with the protocol messages.

In order to formulate our results, we borrow the notion of ideals on strand spaces [11], and we show how this concept is useful in a trace model context for stating and proving secrecy invariants. We show how the complement of an ideal, which we call a *coideal*, serves as a catalyst to apply Paulson's calculus-like set operators. Our protocol model is also unusual in that message events are interspersed with "spell" events that generate the short-term secrets in a session and specify which principals are supposed to share them.

We originally intended this investigation to support our work in applying a theorem prover to inductive protocol proofs using PVS [8, 6]. However, we discovered that these techniques are so effective that we could perform some proofs by hand using them. Manual proofs have been done before, such as the strand-space proofs in [11] and Schneider's CSP proofs [10], but not for Paulson's trace model, and not for a difficult protocol like Lowe's corrected version of the Needham-Schroeder public key protocol [4]. Examples of secrecy proofs are included here for that protocol and for the Otway-Rees [7] protocol.

2 The Modeling Approach

Our modeling approach closely follows Paulson's [9], although the details of the notation are different. A protocol is a rule for adding messages and other events to a history or trace of past events, represented as a sequence. Encrypted message fields are represented symbolically by terms indicating the key and plaintext field.

2.1 Fields

The modeling task begins by defining the *primitive* data types that may occur as message fields: agents $\mathcal{A}$, keys $\mathcal{K}$, and nonces $\mathcal{N}$. (In another context we might use "principal" instead of "agent.") These sets are assumed to be disjoint, and they are all subtypes (subsets) of the field type $\mathcal{F}$. As a notational convention, variables A, B and variants always stand for agents; K and variants always stand for keys; and N and variants are always nonces. X, Y, and Z are arbitrary fields.

Each agent A has some *long-term keys*: a public key $\mathsf{pub}(A)$, a corresponding private key $\mathsf{prv}(A)$, and a symmetric key $\mathsf{shr}(A)$. The set of long-term keys is denoted $\mathcal{K}_L$. Since $\mathsf{pub}(A)$ is intended to be public, we define the *long-term secrets* as a subset of $\mathcal{K}_L$, namely:

$$\mathcal{K}_{LS} = \{\mathsf{prv}(A) | A \in \mathcal{A}\} \cup \{\mathsf{shr}(A) | A \in \mathcal{A}\}.$$

The set of *short-term keys* is denoted $\mathcal{K}_S$. We assume that short-term keys are symmetric.

The *basic* fields are those in the set $\mathcal{N} \cup \mathcal{K}$. These are the kinds of primitive fields that may be designated as secret according to the policy that the protocol is supposed to uphold. Agents and compound fields are never designated as secret by policy, though some compound fields may have to be protected to maintain the secrecy of some of their components.

Compound fields are constructed by concatenation or encryption. The concatenation of X and Y is the term X, Y. We will add brackets, as $[X, Y]$, when necessary to separate a concatenation from its context to avoid confusion. The concatenation operator is binary but associative, so that it may be viewed as n-ary, and a term like $[X, Y, Z]$ is unambiguous.

The encryption of X using the key K is $\{X\}_K$, regardless of the type of key. Each key K has an inverse K^{-1} such that anyone holding $\{X\}_K$ and K^{-1} can compute X. If K is symmetric, $K^{-1} = K$. For any agent A,

$$\begin{aligned} \mathsf{pub}(A)^{-1} &= \mathsf{prv}(A), \\ \mathsf{prv}(A)^{-1} &= \mathsf{pub}(A), \\ \mathsf{shr}(A)^{-1} &= \mathsf{shr}(A). \end{aligned}$$

There are two special agents: Srv, a trusted server assumed to hold the symmetric (and thus, shared) key $\mathsf{shr}(A)$ of any agent A; and the intruder Spy.

Given sets $\mathcal{A}$, $\mathcal{K}$, and $\mathcal{N}$, and the operators whose signatures and relations have just been given for them, there is an initial algebra generated by them [5]; this algebra is the *cryptospace* of fields. It is an idealized abstraction of the true set of message fields in several ways. For example, there may be an infinite number of nonces and keys, and repeated encryption with the same key generates an infinite number of values.

2.2 Events

There are two kinds of events: messages and spells. Messages are essentially Paulson's Says events, and spells may be thought of as a variation on the Notes event, but with a different purpose.

A *message* is an event $A \to B : X$, where (as implied by our notational conventions) A and B are agents, and X is a field. The *content* X of a message event M is denoted by $\underline{M}$. The sender A and the receiver B will always be the *true* sender and intended receiver, as in Paulson's model.

A *spell* generates certain session-specific primitive fields and designates them as secret. A spell is an event $S \ddagger L$, where S is a set of short-term basic fields called the *book*, and L, the so-called *cabal*, is a set of agents who are permitted to share the secrets in S. The book and the cabal of a spell event C are denoted by C_σ and C_α, respectively.

As a notational convention, we use E (and variants) to denote events, while M is a message and C is a spell.

A *trace* is a finite sequence of events. Notationally, variants of H are traces. We indicate trace concatenation or postfixing an event to a trace with juxtaposition, e.g., HE, and $E \in H$ means that E occurs in the sequence H, so that $H = H'EH''$. The empty trace is ϵ.

We extend the notion of a content to traces in the natural way. Spells do not contribute to the content.

$$\underline{H} = \{\underline{M} | M \in H\}.$$

2.3 Inductive Relations

The fundamental operations on sets S of message fields, as introduced by Paulson, are $\mathsf{parts}(S)$, $\mathsf{analz}(S)$, and $\mathsf{synth}(S)$.

Briefly, $\mathsf{parts}(S)$ is the set of all subfields of fields in the set S, including components of concatenations and the plaintext of encryptions (but not the keys). Note that if $X \in \mathsf{parts}(\{Y\})$ then X is a subterm of Y, in the sense of [11], written $X \sqsubseteq Y$. The subterm relation is a partial order.

$\mathsf{analz}(S)$ is the subset of $\mathsf{parts}(S)$ consisting of only those subfields that are accessible to an attacker. These include components of concatenations, and the plaintext of those encryptions where the inverse key is in $\mathsf{analz}(S)$. Thus, $\mathsf{analz}(S)$ is defined to be the smallest set such that

1. $S \subseteq \mathsf{analz}(S)$
2. if $[X, Y] \in \mathsf{analz}(S)$ then $X \in \mathsf{analz}(S)$ and $Y \in \mathsf{analz}(S)$
3. if $\{X\}_K \in \mathsf{analz}(S)$ and $K^{-1} \in \mathsf{analz}(S)$ then $X \in \mathsf{analz}(S)$.

Finally, $\mathsf{synth}(S)$ is the set of fields constructible from S by concatenation and encryption using fields and keys in S. Thus, $\mathsf{synth}(S)$ is defined to be the smallest set such that

1. $S \subseteq \mathsf{synth}(S)$
2. if $X \in \mathsf{synth}(S)$ and $Y \in \mathsf{synth}(S)$ then $[X, Y] \in \mathsf{synth}(S)$
3. if $X \in \mathsf{synth}(S)$ and $K \in \mathsf{synth}(S)$ then $\{X\}_K \in \mathsf{synth}(S)$.

The following properties are stated, for similarly defined sets, in [9]. They are all proved by straightforward inductions.

Proposition 2.1 *The set transformers* $\mathsf{parts}(S)$*,* $\mathsf{analz}(S)$*, and* $\mathsf{synth}(S)$ *are* closure *operators – that is, they are extensive* ($S \subseteq \mathsf{parts}(S)$)*, monotonic, and idempotent. Furthermore:*

$$\begin{array}{rcl} \mathsf{parts}(\mathsf{analz}(S)) & = & \mathsf{parts}(S) \\ \mathsf{analz}(\mathsf{parts}(S)) & = & \mathsf{parts}(S) \\ \mathsf{parts}(\mathsf{synth}(S)) & = & \mathsf{parts}(S) \cup \mathsf{synth}(S) \\ \mathsf{analz}(\mathsf{synth}(S)) & = & \mathsf{analz}(S) \cup \mathsf{synth}(S) \end{array}$$

The intruder in our model synthesizes faked messages from analyzable parts of a set of available fields. This motivates the definition of $\mathsf{fake}(S)$.

Definition 2.1 $\mathsf{fake}(S) = \mathsf{synth}(\mathsf{analz}(S))$

Some useful properties are easy to obtain:

Lemma 2.2 (Fake-Parts)

1. $\mathsf{parts}(\mathsf{fake}(S)) = \mathsf{parts}(S) \cup \mathsf{fake}(S)$

2. $\mathsf{parts}(\mathsf{fake}(S)) \subseteq \mathsf{synth}(\mathsf{parts}(S))$

Proof. Using the equalities in Proposition 2.1:

$$\begin{array}{rcl} \mathsf{parts}(\mathsf{fake}(S)) & = & \mathsf{parts}(\mathsf{analz}(S)) \cup \mathsf{fake}(S) \\ & = & \mathsf{parts}(S) \cup \mathsf{fake}(S) \\ & = & \mathsf{parts}(S) \cup \mathsf{synth}(\mathsf{analz}(S)) \\ & \subseteq & \mathsf{synth}(\mathsf{parts}(S)) \end{array}$$

since $\mathsf{analz}(S) \subseteq \mathsf{parts}(S)$. □

3 Ideals and Coideals

If the spy ever obtains some secret field X, it can transmit X as the content of a message. Thus, our secrecy policy is that if $A \to B : X$ occurs in some trace, then $X \notin S$, where S is a set of secrets.

The invariant that we will actually prove is that $X \notin \mathcal{I}(S)$, where $\mathcal{I}(S)$ is the *ideal* generated by S: the smallest set of fields that includes S and which is closed under concatenation with any fields and under encryption with keys whose inverses are not in S. $\mathcal{I}(S)$ is the k-ideal $I_k[S]$ from [11] where k is the set of keys whose inverses are not in S.

With our choice of k, the ideal is defined as follows:

Definition 3.1 (Ideal) $\mathcal{I}(S)$ *is the smallest set such that*

1. $S \subseteq \mathcal{I}(S)$

2. if $X \in \mathcal{I}(S)$ *or* $Y \in \mathcal{I}(S)$ *then* $[X, Y] \in \mathcal{I}(S)$

3. if $X \in \mathcal{I}(S)$ *and* $K^{-1} \notin S$ *then* $\{X\}_K \in \mathcal{I}(S)$

Under the assumption that any term not in the ideal may be already compromised, it is necessary to protect this whole ideal, because compromising any element of the ideal effectively compromises some element of S. It turns out that protecting this ideal is also sufficient.

It will be useful to note that any term in the ideal must have a subterm in the generating set S. This is proved by induction, but it is evident from the definition of the ideal.

Lemma 3.1 (Ideal Parts) *If* $X \in \mathcal{I}(S)$ *then there exists* $Z \sqsubseteq X$ *such that* $Z \in S$.

The complement of $\mathcal{I}(S)$, which we call a *coideal*, is denoted by $\mathcal{C}(S)$. The coideal $\mathcal{C}(S)$ defines the set of fields that are *public* with respect to the basic secrets S, i.e., fields whose release would not compromise any secrets in S.

The property that makes the notion of "coideal" worth defining is that coideals are closed under attacker analysis, thereby implying that protection of the ideal is sufficient.

Lemma 3.2 (Analz Closure) *For a set S of fields:*

$$\mathsf{analz}(\mathcal{C}(S)) \quad = \quad \mathcal{C}(S)$$

Proof. The right-to-left inclusion follows from extensivity of $\mathsf{analz}(.)$ (Proposition 2.1). We apply the smallest-set definition of $\mathsf{analz}(.)$ to show

$$\mathsf{analz}(\mathcal{C}(S)) \subseteq \mathcal{C}(S).$$

We have to show that $\mathcal{C}(S)$ is closed under the two rules that expand $\mathsf{analz}(.)$.

First, suppose $[X, Y] \in \mathcal{C}(S)$. That is, $[X, Y] \notin \mathcal{I}(S)$. Hence neither X nor Y is in $\mathcal{I}(S)$ by definition of the ideal, so both are in $\mathcal{C}(S)$.

Second, suppose $\{X\}_K \in \mathcal{C}(S)$ and $K^{-1} \in \mathcal{C}(S)$. $\{X\}_K \notin \mathcal{I}(S)$ implies that either $X \notin \mathcal{I}(S)$ or $K^{-1} \in \mathcal{I}(S)$. The first subcase means $X \in \mathcal{C}(S)$ and the latter subcase contradicts the assumption $K^{-1} \in \mathcal{C}(S)$. □

An analogous result does not hold for synthesis in general, but it does hold when the elements generating the coideal are primitive.

Lemma 3.3 (Synth Closure) *For a set S of primitive fields:*

$$\mathsf{synth}(\mathcal{C}(S)) \quad = \quad \mathcal{C}(S)$$

Proof. The left-to-right inclusion is extensivity (Proposition 2.1). So it remains to show

$$\mathsf{synth}(\mathcal{C}(S)) \subseteq \mathcal{C}(S).$$

We must show that $\mathcal{C}(S)$ is closed under the two rules that expand $\mathsf{synth}(.)$.

First, let $X \in \mathcal{C}(S)$ and $Y \in \mathcal{C}(S)$. We must show that $[X, Y] \in \mathcal{C}(S)$. Otherwise, $[X, Y] \in \mathcal{I}(S)$, either because $[X, Y] \in S$ or because X or $Y \in \mathcal{I}(S)$. The former cannot be true because S is primitive and the latter would contradict the hypothesis for this case.

Second, let $X \in \mathcal{C}(S)$ and $K \in \mathcal{C}(S)$. We must show that $\{X\}_K \in \mathcal{C}(S)$. Otherwise, $\{X\}_K \in \mathcal{I}(S)$, either because $\{X\}_K \in S$, not possible for primitive S; or partly because $X \in \mathcal{I}(S)$, which contradicts the hypothesis for this case. □

Lemmas 3.2, 3.3 are typically used to reduce proof obligations like $\mathsf{analz}(T) \subseteq \mathcal{C}(S)$ to $T \subseteq \mathcal{C}(S)$; similarly for $\mathsf{synth}(.)$.

4 Protocols and Secrecy

A protocol specifies which messages or spells can be appended to an event trace. A secret in a spell book must be *unused* in the prior trace, in the sense that it is not a part of any message content and it has not occurred as a secret in a prior spell.

4.1 Basic Definitions

Definition 4.1 (Unused) *If H is a trace, X is* unused *in H if X is basic, $X \notin \mathsf{parts}(\underline{H})$, and $X \notin C_\sigma$ for any $C \in H$. The set of unused fields in H is denoted by* $\mathsf{unused}(H)$.

Definition 4.2 (Protocol) *A* protocol *is a binary relation between traces and events, such that if $(H, C) \in P$ then $C_\sigma \subseteq \mathsf{unused}(H)$.*

A *universe* is a set of traces that could be generated by a protocol in an environment with intruder activity, given some set of fields I assumed initially held by the intruder.

Definition 4.3 (Universe) *If P is a protocol then its* I-universe *$\mathcal{U}_I(P)$ is the set of traces defined inductively by:*

1. *$\epsilon \in \mathcal{U}_I(P)$*
2. *If $H \in \mathcal{U}_I(P)$ and $(H, E) \in P$ then $HE \in \mathcal{U}_I(P)$*
3. *If $H \in \mathcal{U}_I(P)$ and M is a message from* Spy *with $\underline{M} \in \mathsf{fake}(\underline{H} \cup I)$ then $HM \in \mathcal{U}_I(P)$*

A message is called honest *(for $\mathcal{U}_I(P)$) if it has been introduced to a trace by means of rule (2) above, while messages introduced by (3) are* fake.

Because protocol spell books introduce unused secrets, it is easy to show that the spell books of different spells are disjoint.

Lemma 4.1 (Disjoint Book) *If $C, C' \in H \in \mathcal{U}_I(P)$ then either $C = C'$ or $C_\sigma \cap C'_\sigma = \emptyset$.*

The basic secrets associated with a spell include not only the elements of the spell book but also the long-term secrets of the agents in the cabal.

Definition 4.4 (Basic Secrets) *Let C be a spell;*

$$\mathsf{S}_C = C_\sigma \cup \{\mathsf{prv}(A) | A \in C_\alpha\} \cup \{\mathsf{shr}(A) | A \in C_\alpha\}$$

A spell is *compatible* with an initial knowledge set I if I does not contain any of the secrets in the spell's book as subterms.

Definition 4.5 (Compatible Spell) *A spell C is compatible with I if $\mathsf{S}_C \cap \mathsf{parts}(I) = \emptyset$.*

An I-compatible spell is included in the coideal generated by the spell's basic secrets. This follows immediately from the ideal-parts lemma (Lemma 3.1).

Lemma 4.2 (Compatible Coideal) *if C is I-compatible then $I \subseteq \mathcal{C}(\mathsf{S}_C)$.*

The compatible coideal property is important because there is no hope of protecting secrets in a spell that have already been given away to the intruder in the initial knowledge. The definition of compatibility is somewhat stronger than this in order to make use of regularity, defined below.

4.2 Regularity

Long-term secrets are protected if they are never introduced as parts of any message by an honest agent. Paulson would call this a regularity result. We define a protocol to be regular if it protects long-term secrets this way.

Definition 4.6 (Regularity) *A protocol P is* regular *if*

1. $(H, M) \in P$
2. $K \in \mathcal{K}_{LS}$
3. $K \notin \mathsf{parts}(\underline{H})$

implies that $K \notin \mathsf{parts}(\underline{M})$.

Regularity is easy to check from a protocol specification, because one need only look to see if any long-term secret is mentioned explicitly in any way other than using it as a key. A protocol whose purpose is to distribute long-term keys would not satisfy this property, and would have to be verified using some other strategy. Regularity does not prevent the attacker from putting its own long-term keys, or a long-term key from its initial knowledge, into a message; such a key could then be forwarded unwittingly (but benignly) by an honest agent.

Regularity is applied with the help of the following lemma.

Lemma 4.3 (Regularity) *Suppose that*

1. *P is regular*
2. $K \in \mathcal{K}_{LS}$
3. $K \notin \mathsf{parts}(I)$
4. $H \in \mathcal{U}_I(P)$

then $K \notin \mathsf{parts}(\underline{H})$.

The regularity lemma is proved by induction on H. The induction step is immediate if the next event is honest, while if the next event is fake, the proof is an exercise in applying the fake-parts lemma (Lemma 2.2).

4.3 The Secrecy Theorem

We now introduce the fundamental secrecy property of a trace with respect to some assumed initial knowledge. A trace is *discreet* for an initial I if it does not expose the basic secrets of any spell compatible with I.

Definition 4.7 (Discreet Trace) *A trace H is* I-discreet *if, for all spell events $C \in H$ that are compatible with I,*

$$\underline{H} \subseteq \mathcal{C}(\mathsf{S}_C)$$

A protocol is secure with respect to its secrecy policy and the spy's initial knowledge if every trace in the universe it generates is discreet. The secrecy proof for a protocol has a protocol-independent part and a protocol-dependent part. The protocol-dependent part is expressed by the *discreet* property defined below. It says that if the prior trace is discreet, the next message event generated by the protocol does not compromise a secret. This has to be proved individually for each protocol.

Definition 4.8 (Discreet Protocol) *A protocol P is* discreet *if, for all H, I, and C satisfying the conditions:*

1. *$C \in H \in \mathcal{U}_I(P)$ such that H is I-discreet,*
2. *$(H, M) \in P$, and*
3. *C is compatible with I*

it is the case that $\underline{M} \subseteq \mathcal{C}(\mathsf{S}_C)$.

The protocol-independent part of a secrecy proof is the Secrecy Theorem.

Theorem 4.4 (Secrecy) *If P is regular and discreet then every trace in $\mathcal{U}_I(P)$ is I-discreet.*

Proof. By induction on the trace H. If $H = \epsilon$ then there is nothing to prove, since H contains no spell.

Consider a trace $HE \in \mathcal{U}_I(P)$. We have $H \in \mathcal{U}_I(P)$ and $(H, E) \in P$. The induction hypothesis is that H is I-discreet. For the induction step, we must show that HE is I-discreet. Choose a spell $C \in HE$ such that C is compatible with I.

There are two cases, depending on whether $C \in H$ or $C = E$. Assume first that $C \in H$. Since H is I-discreet and C is I-compatible, we have $\underline{H} \cup I \subseteq \mathcal{C}(\mathsf{S}_C)$ and by analz closure (Lemma 3.2) we also have

$$\mathsf{analz}(\underline{H} \cup I) \subseteq \mathcal{C}(\mathsf{S}_C).$$

We must show that $\underline{HE} \subseteq \mathcal{C}(\mathsf{S}_C)$. If E is a spell, we are done since $\underline{HE} = \underline{H}$ in that case.

Assume that E is a message, $E = M$. Since $\underline{HM} = \underline{H} \cup \{\underline{M}\}$ it remains only to show that $\underline{M} \in \mathcal{C}(\mathsf{S}_C)$.

The message M might be either honest or fake. In either case $\underline{M} \in \mathcal{C}(\mathsf{S}_C)$. For, if M is honest, this is true because P is discreet. If M is fake, $\underline{M} \in \mathsf{fake}(\underline{H} \cup I)$. By the induction hypothesis, monotonicity of synth, and synth closure (Lemma 3.3), we have

$$\begin{aligned} \mathsf{fake}(\underline{H} \cup I) &= \mathsf{synth}(\mathsf{analz}(\underline{H} \cup I)) \\ &\subseteq \mathsf{synth}(\mathcal{C}(\mathsf{S}_C)) \\ &= \mathcal{C}(\mathsf{S}_C). \end{aligned}$$

We return now to the case where the spell of interest is not in H, but has just been added, so $E = C$. In this case $\underline{HE} = \underline{H}$, but the induction hypothesis does not tell us that $\underline{H} \subseteq \mathcal{C}(\mathsf{S}_C)$, because $C \notin H$.

Instead, we use regularity and other properties of the protocol. Suppose that $X \in \underline{H}$ and $X \in \mathcal{I}(\mathsf{S}_C)$. By the ideal-parts lemma (Lemma 3.1), there exists $Z \sqsubseteq X$ such that $Z \in \mathsf{S}_C$. Note that $Z \in \mathsf{parts}(\underline{H})$.

There are two cases: $Z \in C_\sigma$ or $Z \in \mathcal{K}_{LS}$.

Suppose $Z \in C_\sigma$. Since $(H, C) \in P$, we know that $C_\sigma \subseteq \mathsf{unused}(H)$, so $Z \notin \mathsf{parts}(\underline{H})$, contradiction.

Suppose $Z \in \mathcal{K}_{LS}$. Since C is compatible with I, $\mathsf{S}_C \cap \mathsf{parts}(I) = \emptyset$, so $Z \notin \mathsf{parts}(I)$. By regularity (Lemma 4.3), $Z \notin \mathsf{parts}(\underline{H})$. Again we have a contradiction, and this completes the proof. □

In the following sections we give examples of proofs of the discreet property for two protocols, from which we may conclude, by the secrecy theorem, that their traces are discreet. These are strictly secrecy results, and show only that the secrets generated in a particular run of the protocol are not compromised. Most authors of protocol proofs have noted that the security objectives of a protocol may be undermined in other ways than by compromising secrets, usually due to some failure of authentication. We discuss this concern in the Conclusion.

5 Example: The Otway-Rees Protocol

The Otway-Rees protocol is a good one to begin with because the proof is short. Also this protocol was used as an example in [11], so that one can make a comparison between the effort required here with the effort required to do the secrecy part of the strand-space proof in that paper (which was also fairly concise).

The goal of the Otway-Rees protocol is to mutually authenticate an initiator and responder and to distribute a session key generated by the server. One session consists of the four messages in Figure 1. We prove that none of the secrets N_a, N_b, or K are disclosed.

$$\begin{array}{l} A \to B : N, A, B, \{N_a, N, A, B\}_{\mathsf{shr}(A)} \\ B \to \mathsf{Srv} : N, A, B, \{N_a, N, A, B\}_{\mathsf{shr}(A)}, \\ \qquad\qquad \{N_b, N, A, B\}_{\mathsf{shr}(B)} \\ \mathsf{Srv} \to B : N, \{N_a, K\}_{\mathsf{shr}(A)}, \{N_b, K\}_{\mathsf{shr}(B)} \\ B \to A : N, \{N_a, K\}_{\mathsf{shr}(A)} \end{array}$$

Figure 1: The Otway-Rees Protocol

The informal rules in Figure 1 are easily, albeit somewhat tediously, encoded in the trace model, in roughly the way Paulson would do it, except for the spell event. The spell is specified by g_0 in Definition 5.1, which generates the two nonces N_a, N_b, and the session key K. Note that the server need not be mentioned in the cabal.

The relations or_1, or_2, or_3, or_4 in Definition 5.1, on message events M and traces H, correspond to the messages in the informal description of the protocol. Relation $or_i(H, M)$ holds when rule or_i is used to generate message M. A message is not sent unless there is a suitable prior history of messages sent and received by the sending agent. Rules that introduce nonces take them from a prior spell with the expected cabal. When an agent uses a secret from a spell book, the agent does not see any of the other secrets in the same spell book, though it might know about them from prior messages.

In general, a trace generated by these rules interleaves the behavior of as many agents as we wish, and any number of concurrent or sequential sessions of the same agents. Also, once a message is enabled, it can be added to the trace any number of times. This is unrealistic, but it is a possible consequence of attacker behavior, and it does not affect secrecy conclusions.

In these rules, a condition $M = ...$ indicates the new message, a condition $C \in H$ identifies a prior spell event that specifies session secrets and participating agents, a condition $A \to B... \in H$ identifies a prior message. A prior message might be one that has just been received or one previously sent by the agent sending the new M. Messages previously sent by the same agent are named in order to establish the state of that agent. Note that the sender of a message that is just received is unconstrained and ignored, because the receiver can respond only to the content of the message.

Definition 5.1 (Otway-Rees) *OR is the union of the relations*

$g_0(H, C) = (\exists N_a, N_b, K, A, B)$
$\quad N_a, N_b, K \in \mathsf{unused}(H)$
$\quad \wedge\ C = \{N_a, N_b, K\} \ddagger \{A, B\}$
$or_1(H, M) = (\exists A, B, N, N_a, C)$
$\quad C \in H \wedge N \in \mathsf{unused}(H)$
$\quad \wedge\ N_a \in C_\sigma \wedge C_\alpha = \{A, B\}$
$\quad \wedge\ M = A \to B : N, A, B, \{N_a, N, A, B\}_{\mathsf{shr}(A)}$
$or_2(H, M) = (\exists A, A', B, X, N, N_b, C)$
$\quad C \in H$
$\quad \wedge\ N_b \in C_\sigma \wedge C_\alpha = \{A, B\}$
$\quad \wedge\ A' \to B : N, A, B, X \in H$
$\quad \wedge\ M = B \to \mathsf{Srv} : N, A, B, X, \{N_b, N, A, B\}_{\mathsf{shr}(B)}$
$or_3(H, M) = (\exists A, B, B', N, N_a, N_b, K, C)$
$\quad C \in H$
$\quad \wedge\ K \in C_\sigma \wedge C_\alpha = \{A, B\}$
$\quad \wedge\ B' \to \mathsf{Srv} : N, A, B, \{N_a, N, A, B\}_{\mathsf{shr}(A)},$
$\qquad \{N_b, N, A, B\}_{\mathsf{shr}(B)} \in H$
$\quad \wedge\ M = \mathsf{Srv} \to B : N, \{N_a, K\}_{\mathsf{shr}(A)}, \{N_b, K\}_{\mathsf{shr}(B)}$
$or_4(H, M) = (\exists A, B, N, X, Y, N_b, K)$
$\quad B \to \mathsf{Srv} : N, A, B, X, \{N_b, N, A, B\}_{\mathsf{shr}(B)} \in H$
$\quad \wedge\ \mathsf{Srv} \to B : N, Y, \{N_b, K\}_{\mathsf{shr}(B)} \in H$
$\quad \wedge\ M = B \to A : N, Y$

OR is a protocol in the sense of Definition 4.2, since g_0 only puts previously unused fields into the book, and it is regular by inspection. From the Secrecy Theorem 4.4 and the following lemma it follows that OR is secure.

Theorem 5.1 *The OR protocol is discreet.*

Proof. Let P be OR and choose I. Let $(H, M) \in P$, where $H \in \mathcal{U}_I(P)$ such that H is I-discreet. Let $C \in H$ such that $I \subseteq \mathcal{C}(\mathsf{S}_C)$ and $C_\sigma \cap \mathsf{parts}(I) = \emptyset$. We have to show

$$\underline{M} \subseteq \mathcal{C}(\mathsf{S}_C).$$

There are four message rules.

First, consider the case $(H, M) \in or_1$; then:

$$\begin{aligned} M &= A \to B : N, A, B, \{N_a, N, A, B\}_{\mathsf{shr}(A)} \\ C' &= \{N_a, N_b, K\} \ddagger \{A, B\} \in H \end{aligned}$$

Notice that $N \notin \mathsf{S}_C$, because N is unused, and $A, B \notin \mathsf{S}_C$ because they are agents. Now consider the encrypted term.

Case $A \in C_\alpha$**.** Then $\mathsf{shr}(A) \in \mathsf{S}_C$; so $\underline{M} \notin \mathcal{C}(\mathsf{S}_C)$.

Case $A \notin C_\alpha$**.** Then $C' \neq C$, so $N_a \notin \mathsf{S}_C$. Hence, $\underline{M} \notin \mathcal{C}(\mathsf{S}_C)$.

Second, in case $(H, M) \in or_2$,

$$\begin{aligned} M &= B \to \mathsf{Srv} : N, A, B, X, \{N_b, N, A, B\}_{\mathsf{shr}(B)} \\ M_1 &= A' \to B : N, A, B, X \in H \end{aligned}$$

where $N_b \in C'_\sigma$ and $C'_\alpha = \{A, B\}$. Since $M_1 \in H$ and H is I-discreet, $\mathsf{analz}(\underline{H} \cup I) \subseteq \mathcal{C}(\mathsf{S}_C)$. But the unencrypted terms $N, A, B, X \in \mathsf{analz}(\underline{M_1}) \subseteq \mathsf{analz}(\underline{H} \cup I)$ so $N, A, B, X \in \mathcal{C}(\mathsf{S}_C)$. The encrypted term is also in the coideal, using the same arguments as for or_1.

Third, in case $(H, M) \in or_3$,

$$\begin{aligned} M &= \mathsf{Srv} \to B : N, \{Na, K\}_{\mathsf{shr}(A)}, \{N_b, K\}_{\mathsf{shr}(B)} \\ M_2 &= B' \to \mathsf{Srv} : N, A, B, \{N_a, N, A, B\}_{\mathsf{shr}(A)}, \\ &\qquad \{N_b, N, A, B\}_{\mathsf{shr}(B)} \in H \end{aligned}$$

and there exists a spell $C' \in H$ such that $K \in C'_\sigma$ and $C'_\alpha = \{A, B\}$.

We know $N \in \mathcal{C}(\mathsf{S}_C)$ because it came from M_2. The first encrypted term of M is in the coideal if $A \in C_\alpha$. Otherwise, assume $A \notin C_\alpha$. We know $C \neq C'$ and we must consider the components N_a, K. $K \notin C_\sigma$, so $K \in \mathcal{C}(\mathsf{S}_C)$. As for N_a, it comes from a term in M_2 encrypted with $\mathsf{shr}(A)$, so $N_a \in \mathcal{C}(\mathsf{S}_C)$ because H is I-discreet.

The same argument can be used for the second encrypted term and N_b.

The fourth case $(H, M) \in or_4$ is trivial because the message fields have been copied from a received message in H, which is I-discreet. □

6 Example: The Needham-Schroeder Public-Key Protocol

The Needham-Schroeder public-key protocol is a more challenging example, which to our knowledge has not been verified by hand before. The original protocol was found to be flawed by Lowe, who suggested a change in one message that made it secure, as far as he could tell from a model-checking analysis [4]. We demonstrate the applicability of the secrecy theorem (Theorem 4.4) by proving that Lowe's corrected version of the protocol, which we refer to as NSL, is secure.

$$\begin{aligned} &A \to B : \{N_a,\, A\}_{\mathsf{pub}(B)} \\ &B \to A : \{N_a,\, N_b,\, B\}_{\mathsf{pub}(A)} \\ &A \to B : \{N_b\}_{\mathsf{pub}(B)} \end{aligned}$$

Figure 2: The Needham-Schroeder-Lowe Protocol

The informal description of the NSL protocol is in Figure 2. Here is the trace relation version.

Definition 6.1 (NSL Protocol) *The protocol NSL is defined as the union of the binary relations g_0, a_1, b_2, and a_3.*

$$\begin{aligned}
&g_0(H, C) = (\exists A, B, N_a, N_b) \\
&\quad N_a, N_b \in \mathsf{unused}(H) \\
&\quad \wedge\ C = \{N_a, N_b\} \ddagger \{A, B\} \\
&a_1(H, M) = (\exists A, B, C, N_a) \\
&\quad C \in H \wedge\ N_a \in C_\sigma \wedge\ C_\alpha = \{A, B\} \\
&\quad \wedge\ M = A \to B : \{N_a,\, A\}_{\mathsf{pub}(B)} \\
&b_2(H, M) = (\exists A, B, A', C, N_a, N_b) \\
&\quad C \in H \wedge\ N_b \in C_\sigma \wedge\ C_\alpha = \{A, B\} \\
&\quad \wedge\ A' \to B : \{N_a,\, A\}_{\mathsf{pub}(B)} \in H \\
&\quad \wedge\ M = B \to A : \{N_a,\, N_b,\, B\}_{\mathsf{pub}(A)} \\
&a_3(H, M) = (\exists A, B, B', N_a, N_b) \\
&\quad A \to B : \{N_a,\, A\}_{\mathsf{pub}(B)} \in H \\
&\quad \wedge\ B' \to A : \{N_a,\, N_b,\, B\}_{\mathsf{pub}(A)} \in H \\
&\quad \wedge\ M = A \to B : \{N_b\}_{\mathsf{pub}(B)}
\end{aligned}$$

It is immediate from the definition of NSL that secrets in spells are unused in the prior trace, thus NSL is a protocol in the sense of Definition 4.2. It is regular by inspection.

Theorem 6.1 (NSL Protection) *The NSL protocol is discreet.*

Proof. Let P be the NSL protocol and choose I. Let $(H, M) \in P$, where $H \in \mathcal{U}_I(P)$ such that H is I-discreet. Let $C \in H$ such that $I \subseteq \mathcal{C}(\mathsf{S}_C)$ and $C_\sigma \cap \mathsf{parts}(I) = \emptyset$. Since H is I-discreet, $\mathsf{analz}(\underline{H} \cup I) \subseteq \mathcal{C}(\mathsf{S}_C)$. We have to show that $\underline{M} \subseteq \mathcal{C}(\mathsf{S}_C)$.

There are three message rules.

Case 1. $a_1(H, M)$.

$$M \quad = \quad A' \to B : \{N_a,\, A\}_{\mathsf{pub}(B)}$$

and $C' \in H$ such that $N_a \in C'_\sigma$ and $C'_\alpha = \{A, B\}$. If $B \in C_\alpha$ then $\mathsf{shr}(B) \in \mathsf{S}_C$ and the encrypted term is in the coideal. Otherwise, assume $B \notin C_\alpha$. Then $C \neq C'$ and $N_a \notin C_\sigma$. This fact, together with $A \notin \mathcal{I}(\mathsf{S}_C)$ yields $\underline{M} \notin \mathcal{I}(\mathsf{S}_C)$.

Case 2. $b_2(H, M)$.

$$M = B \to A : \{N_a,\, N_b,\, B\}_{\mathsf{pub}(A)}$$

and there must exist

$$M_1 = A' \to B : \{N_a,\, A\}_{\mathsf{pub}(B)} \in H$$

and $C' \in H$ such that $N_b \in C'_\sigma$ and $C'_\alpha = \{A, B\}$.

If $A \in C_\alpha$ then $\underline{M} \notin \mathcal{I}(\mathsf{S}_C)$ and we are done. Suppose $A \notin C_\alpha$. Then we must show that N_a, N_b, and $B \in \mathcal{C}(\mathsf{S}_C)$. N_b is handled like N_a in the first message and B is an agent. It is also trivial for N_a if $B \notin C_\alpha$ because N_a is then exposed in M_1 and H is I-discreet.

We must show that $N_a \notin \mathsf{S}_C$ if we assume that $B \in C_\alpha$. Find the earliest occurrence of the subterm $M_1 =$

$\{N_a, A\}_{\mathsf{pub}(B)}$. That is, there is a message M' whose content has M_1 as a subterm, and M_1 is not a part of the prior trace H'. Also, $M_1 \notin \mathsf{parts}(I)$ unless $N_a \in \mathsf{parts}(I)$, in which case $Na \notin \mathsf{S}_C$ by choice of C.

M' might be either faked or honest. If M' is faked, $M_1 \in \mathsf{parts}(\mathsf{fake}(\underline{H'} \cup I)) = \mathsf{parts}(\underline{H'} \cup I) \cup \mathsf{fake}(\underline{H'} \cup I)$ so that we must have $M_1 \in \mathsf{fake}(\underline{H'} \cup I)$. Since $M_1 \notin \mathsf{parts}(\underline{H'} \cup I)$ it must have been synthesized, meaning $[N_a, A] \in \mathsf{fake}(\underline{H'} \cup I) \subseteq \mathcal{C}(\mathsf{S}_C)$, so $N_a \notin \mathsf{S}_C$.

If M' is honest, inspection of the rules and the message component types shows that $\underline{M'} = M_1$ and $a_1(H', M')$ holds. But the analysis of rule a_1 has already been covered in the first case.

Case 3. $a_3(H, M)$. If the receiving agent B of M is in the cabal, then the content is not in the ideal. Thus, assume that B is not in the cabal. From the definition of a_3,

$$\begin{aligned} M_1 &= A \to B : \{N_a, A\}_{\mathsf{pub}(B)} \in H \\ M_2 &= B' \to A : \{N_a, N_b, B\}_{\mathsf{pub}(A)} \in H \\ M &= A \to B : \{N_b\}_{\mathsf{pub}(B)} \end{aligned}$$

and one has to show $N_b \notin \mathsf{S}_C$.

If M_2 is honest, then there exists a prefix H' of H such that $b_2(H', M_2)$ and there exists C' with $N_b \in C'_\sigma$ and $C'_\alpha = \{A, B\}$. But $B \notin C_\alpha$, so $C \neq C'$ and $N_b \notin \mathsf{S}_C$.

Note that this step fails if the sender of M_2 does not occur in the encryption field, since then we could not say (in the rule) that $B \in C'_\alpha$. This is the difference between NSL and the original protocol.

If M_2 is faked, find the earliest message M' containing M_2 as a part, where the prior trace is H'. By choice of M', $M_2 \notin \mathsf{parts}(\underline{H'})$. Also, $M_2 \notin \mathsf{parts}(I)$, otherwise $N_b \in \mathsf{parts}(I)$, and we have assumed $\mathsf{parts}(I) \cap \mathsf{S}_C = \emptyset$, so $N_b \notin \mathsf{S}_C$ and we would be done. So $M_2 \notin \mathsf{parts}(\underline{H'} \cup I)$.

If M' is faked, we have

$$\begin{aligned} M_2 &\in \mathsf{parts}(\mathsf{fake}(\underline{H'} \cup I)) \\ &= \mathsf{parts}(\underline{H'} \cup I) \cup \mathsf{fake}(\underline{H'} \cup I) \end{aligned}$$

so

$$M_2 \in \mathsf{fake}(\underline{H'} \cup I) = \mathsf{synth}(\mathsf{analz}(\underline{H'} \cup I)).$$

Since $M_2 \notin \mathsf{parts}(\underline{H'} \cup I)$ it must be that M_2 has been synthesized, so

$$N_a, N_b, B \in \mathsf{fake}(\underline{H'} \cup I).$$

But $\mathsf{fake}(\underline{H'} \cup I) \subseteq \mathcal{C}(\underline{H'} \cup I)$ since H is I-discreet, implying that $N_b \notin \mathsf{S}_C$.

If M' is honest, inspection of the protocol rules shows that $M' = M_2$ and $b_2(H, M_2)$, and this case was covered previously. $\square$

7 Conclusions

Our secrecy theorem separates protocol-dependent and protocol-independent aspects of secrecy proofs. The protocol-dependent part is to show the "discreet" property, which only asks whether honest messages compromise secrets, given strong assumptions about the preservation of secrecy in the prior message history.

The secrets to be protected are defined in an explicit, uniform way by introducing "spell" events into the protocol. Spell events generate the short-term secrets for a particular "cabal," the set of agents sharing the new secrets. Secrets are shown to be protected even when the long-term secrets of other agents, or the short-term secrets in other protocol runs (with other spells) are compromised. Spell events serve to coordinate the static quality of ideals with the dynamic policy of protecting short-term secrets in the context of an arbitrary choice of initial knowledge.

The notion of adding special kinds of events that are significant for the security policy is not a new one. Lowe, for example, added a special "commit" event to his FDR model of a protocol to use in identifying a kind of authentication failure [4].

The security of a protocol can be subverted even when the secrets it generates are protected. To take a simple example, consider the single-message key-distribution protocol:

$$A \to B : A, \{K\}_{\mathsf{pub}(B)}.$$

We can show that the session key K is kept secret. However, B would be foolish to believe that K came from A and use it to encrypt information to be shared only with A.

There are two ways to avoid this kind of problem. One is to conduct a separate authentication proof, and attempt to establish that the key received by B was actually sent by A, and is fresh. If *both* the secrecy proof and the authentication proof succeed (and the second will fail in this example), the protocol would be shown secure. While most authors who have developed analysis techniques for secrecy have extended those techniques to perform authentication proofs as well, we should consider that there are some very appealing authentication logic techniques designed for this purpose [1, 3]. Their only drawback is that they cannot show secrecy properties. It would be ideal to use them in a context where secrecy has already been shown.

If the ultimate objective is really to show secrecy, not for the session key *per se,* but for some text encrypted with it, then there is another way to focus on the correct goal: include the use of the key in the protocol. To do this, add one or more statements to the protocol specification. In the example above, we could add the message:

$$B \to A : \{N\}_K$$

where N is a new secret. The augmented protocol is not discreet.

The closure results on the coideal have turned out to be a useful addition to the arsenal of proof techniques, enabling interesting examples to be shown secure. Protocol proofs are still complex enough so that we feel proof-checking and automation to be valuable for the sake of assurance, and we believe that the same techniques that simplify manual proofs will also be helpful in organizing machine-assisted proofs.

References

[1] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, 1990.

[2] N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. In *Formal Methods and Security Protocols*, Federated Logic Conference, 1999.

[3] L. Gong, R. Needham, and R. Yahalom. Reasoning about belief in cryptographic protocols. In *IEEE Symposium on Research in Security and Privacy*, pages 234–248. IEEE Computer Society, 1990.

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

[5] J. Meseguer and J. Goguen. Initiality, induction, and computability. In M. Nivat and J. Reynolds, editors, *Algebraic Methods in Semantics*, pages 459–541. Cambridge University Press, 1982.

[6] J. Millen. A necessarily parallel attack. In *FLoC Workshop on Formal Methods and Security Protocols*, 1999.

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

[8] S. Owre, J. M. Rushby, and N. Shankar. PVS: A prototype verification system. In Deepak Kapur, editor, *11th International Conference on Automated Deduction (CADE)*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 748–752. Springer-Verlag, 1992.

[9] L. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1):85–128, 1998.

[10] S. Schneider. Verifying authentication protocols in CSP. *IEEE Transactions on Software Engineering*, 24(9):741–758, September 1998.

[11] J. Thayer, J. Herzog, and J. Guttman. Honest ideals on strand spaces. In *11th IEEE Computer Security Foundations Workshop*, pages 66–78. IEEE Computer Society, 1998.