Proving secrecy is easy enough

Véronique Cortier
Laboratoire Spécification et Vérification
Ecole Normale Supérieure de Cachan
61, Avenue du Président Wilson,
94230 Cachan, France
cortier@lsv.ens-cachan.fr

Jon Millen and Harald Rueß
SRI International, Computer Science Laboratory
333 Ravenswood Ave, Menlo Park,
CA 94035, USA
{millen,ruess}@csl.sri.com

Abstract

We develop a systematic proof procedure for establishing secrecy results for cryptographic protocols. Part of the procedure is to reduce messages to simplified constituents, and its core is a search procedure for establishing secrecy results. This procedure is sound but incomplete in that it may fail to establish secrecy for some secure protocols. However, it is amenable to mechanization, and it also has a convenient visual representation. We demonstrate the utility of our procedure with secrecy proofs for standard benchmarks such as the Yahalom protocol.

1 Introduction

Cryptographic protocols are used to achieve goals like authentication and key distribution in a possibly hostile environment. These protocols are notoriously difficult to design and test, and serious flaws have been found in many protocols. Consequently there has been a growing interest in applying formal methods for validating cryptographic protocols. In particular, standard program verification techniques such as model checking, theorem proving, or invariant generation have been found to be essential tools; a recent overview has been given by Meadows [Mea00].

A popular choice is to use model checking procedures for debugging purposes by searching for attacks. These techniques, however, are not directly applicable for verification, since search spaces usually can not be explored exhaustively. In contrast, approaches based on theorem proving techniques aim at mathematical proofs of the desired protocol properties [DS97, Sch98, Pau98, Coh00]. We review the techniques that are most closely related to our work. Paulson [Pau98] uses an interactive theorem prover to prove invariance properties by proving that they are *inductive*, i.e. they are preserved by the execution of each and every protocol rule. Domain-specific tactics are crucial for mechanizing the process of proof construction, but verifications still require considerable effort and insight into the workings of the protocol under consideration. Cohen's [Coh00] approach is much more automatic. He constructs a first-order invariant from a protocol description, and uses first-order reasoning for establishing safety properties. Cohen's approach is amenable to both hand proofs and automation. Indeed, he applies his method to verify the large majority of the benchmark protocols in the Clark and Jacob survey [CJ97] with only a small amount of user intervention.

In contrast to the work by Paulson and Cohen we do not consider safety properties in general, but we restrict ourselves to the specific case of proving secrecy invariants of cryptographic protocols; that is, our main interest is in proving that secrets are not accidentally revealed to unauthorized agents. Proving secrecy invariants for cryptographic protocols has often been found to be the hardest task in analyzing a protocol [Pau98]. Indeed, secrecy has been shown to be undecidable even under very weak assumptions on the protocol [DLMS99], while specialized logics for establishing authentication are usually decidable [Mon99]. Authentication logics depend on assumptions about secrecy, so the most efficient way to prove authentication may be to use a technique like ours to prove secrecy, and use a quick decision procedure to establish authentication once the secrecy assumptions have been verified.

Our proof technique is to perform inductive proofs, as advocated by Paulson [Pau98]. To help express secrecy goals, we make use of the "spell" events introduced in [MR00]. However, this paper does not use the trace model as in [MR00] or [Pau98], but a new state-transition model similar to the MSR model proposed by Mitchell *et al* [CDL$^+$99]. We have found that the Secrecy Theorem in [MR00] could be adapted to work just as well in this context. The current model and our use of PVS to perform inductive proofs with this model were presented at a workshop that did not have a published proceedings [RM00]. We have used this approach mainly for proving secrecy of standard benchmark protocols such as the Otway-Rees and the Needham-Schroeder protocol. Dutertre *et al* [DSS01] used our techniques for verifying the group management services of Enclaves [Gon97].

The starting point for this paper is the observation that secrecy proofs based on the decomposition of the Secrecy Theorem follow a standard pattern that is amenable to mechanization, and which also has a convenient visual representation. Part of the procedure is to reduce messages to simplified constituents called *branches*. The core is a search procedure for establishing secrecy results. This procedure is sound and automatic but incomplete in that it may fail to establish secrecy for some secure protocols.

The paper is structured as follows. In Section 2 we review a state-based model for modeling cryptographic protocols, and we state a suitable security policy together with a corresponding secrecy theorem as introduced in [RM00]. This theorem reduces secrecy proofs to local proof obligations on protocol transitions; these obligations are called *occultness* conditions. In Section 3 we develop a characterization of the occultness notion, which is used to initialize our proof procedure. Then, in Section 4 we describe a search procedure for establishing secrecy results. Moreover, Section 4 contains a soundness result for this search procedure, and we sketch a convenient graphical representation for occultness proofs. Section 5 includes some case studies drawn from the Clark-Jacob survey [CJ97] such as the Yahalom and the Kao Chow repeated authentification protocol. We also demonstrate a proof of non-occultness from a failed proof attempt. Section 6 contains some concluding remarks.

2 Background

We give an overview of state-based encodings of protocols, a security policy based on the notion of coideals, and a secrecy theorem for generating local verification conditions. More detailed descriptions can be found in [MR00, RM00].

Message Fields. The set $Fields$ of message fields is made up of primitive and compound fields. The primitive fields are those of types $Agent$, Key, and $Nonce$. Keys and nonces form the set $Basic$; these basic fields are the only types of fields that may be designated as secret, as a protocol policy goal. Compound fields are constructed by concatenation $[X, Y]$ (often written without brackets) or encryption $\{X\}_K$.

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. The reserved subscript "s" identifies a set, so N_s is a set of nonces.

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 $\mathsf{shr}(A)$, which is shared between A and a designated server agent Srv. Each key K has an inverse key K^{-1}; in particular, $\mathsf{pub}(A)^{-1} = \mathsf{prv}(A)$, $\mathsf{prv}(A)^{-1} = \mathsf{pub}(A)$, while $\mathsf{shr}(A)^{-1} = \mathsf{shr}(A)$, as is the case with any symmetric keys. Keys generated during a protocol session are always symmetric keys.

Events and Global States. There are three kinds of events: *message*, *spell*, and *state* events. A message event is simply a field representing the content of the message. A spell event $C = S \ddagger L \in Spells$, generates the *book*, or session-specific set of basic secrets $Book(C) = S$, which are shared among the set $Cabal(C) = L$ of agents, the *cabal*.

A state event is of the form $Q = \mathsf{A}_n(X) \in States$ where A is a role name, n is a natural number that represents the step of the protocol, and $X = Mem(Q)$ is a concatenated field that represents the memory held by the state. We also write $Mem(H) = \{Mem(Q) | Q \in H \cap States\}$ for any event set H. As a notational convention, we use Q (and variants) to denote state events, while M is a message event, and C is a spell event. The set of *basic secrets* of a spell consists of its book plus the long-term keys of its cabal:

$$Sec(C) = Book(C) \cup ltk(Cabal(C))$$

The long-term keys are those generated by $\mathsf{pub}(.)$, $\mathsf{prv}(.)$, and $\mathsf{shr}(.)$.

A *global state* is a set (not a multiset) of events. Notationally, variants of H are global states or event sets. The *content* of a global state is its set of messages, written:

$$Cont(H) \stackrel{\text{def}}{=} H \cap Fields$$

Similarly, the secrets of a global state are obtained from its spell events.

$$Sec(H) \stackrel{\text{def}}{=} \bigcup \{Sec(C) | C \in H\}$$

A basic field X is *unused* in H if it is neither a part of a field in the content nor a secret of H.

$$\begin{aligned} unused(H) \stackrel{\text{def}}{=} \{X \in Basic \mid \\ X \notin \mathsf{parts}(Cont(H)), X \notin Sec(H)\} \end{aligned}$$

Inductive Relations. $\mathsf{parts}(S)$ is the set of all subfields of fields in the set of fields S, including components of concatenations and the plaintext of encryptions (but not the keys). $\mathsf{analz}(S)$ 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 $\mathsf{analz}(S)$.

$$\emptyset \xrightarrow{\{N_a, N_b, K\}} \left\{ \begin{array}{l} \{N_a, N_b, K\} \ddagger \{A, B\}, \\ \mathsf{A}_1(A, B, Srv), \\ \mathsf{B}_1(B, Srv), \\ \mathsf{Srv}_1(Srv) \end{array} \right\} \quad (0)$$

$$\left\{ \begin{array}{l} \{N_a, \_\} \ddagger \{A, B\} \\ \mathsf{A}_1(A, B, Srv) \end{array} \right\} \xrightarrow{\{N\}} \left\{ \begin{array}{l} \mathsf{A}_2(A, B, Srv, N_a), \\ [N, A, \{N_a, N, B\}_{\mathsf{shr}(A)}] \end{array} \right\} \quad (1)$$

$$\left\{ \begin{array}{l} [N, A, X], \\ \mathsf{B}_1(B, Srv), \\ \{N_b, \_\} \ddagger \{A, B\} \end{array} \right\} \xrightarrow{\emptyset} \quad (2)$$
$$\left\{ \begin{array}{l} \mathsf{B}_2(B, Srv, A, N_b), \\ [N, B, A, X, \{N_b, N, A\}_{\mathsf{shr}(B)}] \end{array} \right\}$$

$$\left\{ \begin{array}{l} M, \\ \mathsf{Srv}_1(Srv), \\ \{K, \_\} \ddagger \{A, B\} \end{array} \right\} \xrightarrow{\emptyset} \quad (3)$$
$$\left\{ \begin{array}{l} \mathsf{Srv}_2(Srv), \\ [N, \{N_a, K\}_{\mathsf{shr}(A)}, \{N_b, K\}_{\mathsf{shr}(B)}] \end{array} \right\}$$

$$\{[N, X, \{N_b, K\}_{\mathsf{shr}(B)}]\} \xrightarrow{\emptyset} \{[N, X]\} \quad (4)$$

where $M \stackrel{\text{def}}{=} [N, B, A, \{N_a, N, B\}_{\mathsf{shr}(A)}, \{N_b, N, A\}_{\mathsf{shr}(B)}]$

Figure 1. Encoding of part of Otway-Rees protocol.

More precisely, $\mathsf{analz}(S)$ is the smallest superset of S such that $X \in \mathsf{analz}(S)$ and $Y \in \mathsf{analz}(S)$ if $[X, Y] \in \mathsf{analz}(S)$, and $X \in \mathsf{analz}(S)$ if $\{X\}_K \in \mathsf{analz}(S)$ and $K^{-1} \in \mathsf{analz}(S)$. Finally, $\mathsf{synth}(S)$ is the set of fields constructible from S by concatenation and encryption using fields and keys in S. It is defined to be the smallest superset of S such that $[X, Y] \in \mathsf{synth}(S)$ if $X \in \mathsf{synth}(S)$ and $Y \in \mathsf{synth}(S)$, and $\{X\}_K \in \mathsf{synth}(S)$ if $X \in \mathsf{synth}(S)$ and $K \in \mathsf{synth}(S)$. The intruder in our model synthesizes faked messages from analyzable parts of a set of available fields. This motivates the definition $\mathsf{fake}(S) \stackrel{\text{def}}{=} \mathsf{synth}(\mathsf{analz}(S))$.

Ideals and Coideals. An *ideal* $\mathcal{I}(S)$ denotes the set of fields that have to be protected in order not to reveal any secrets in S [THG98]. It is defined as the smallest superset of S such that $[X, Y] \in \mathcal{I}(S)$ if $X \in \mathcal{I}(S)$ or $Y \in \mathcal{I}(S)$, and $\{X\}_K \in \mathcal{I}(S)$ if $X \in \mathcal{I}(S)$ and $K^{-1} \notin \mathcal{I}(S)$. The complement of an ideal, the coideal, is denoted by $\mathcal{C}(S)$. This 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. Coideals are interesting because they are closed under attacker analysis; i.e. $\mathsf{fake}(\mathcal{C}(S)) = \mathcal{C}(S)$ for all primitive fields S.

Protocols. A protocol transition t is of the form $Pre(t) \xrightarrow{New(t)} Post(t)$, where $Pre(t)$ and $Post(t)$ are set of events and $New(t)$ is a set of nonces. Such transitions specify a possible global state change in a way to be explained below.

Except for an initialization transition, in which $Pre(t)$ is empty, a transition t shows a state change for one role. It may also produce, in the post, a message or a spell but not both. A primitive field occurring in post messages or state memory must occur in the messages or state memory of the pre or among the nonces. This condition is called *regularity,* and it implies that no long-term keys are deliberately introduced into a post message. There is also a restriction that secrets in a post spell are all in $New(t)$. (The freshness of nonces in $New(t)$ is implied by the definition of global state transitions given below.)

A *protocol* is simply a set of protocol transitions. A protocol specification is a set of rules, where each rule is a schema defining a set of transitions using terms with free variables. More formally, a transition t is an *instance* of a rule rl iff there exists a ground substitution σ, defined on the variables of rl, such that $t = \sigma(rl)$.

Protocol rules for the first three messages of the familiar Otway-Rees [OR87] (OR) and Needham-Schroeder-Lowe [NS78, Low96] (NSL) public key protocols can be found in Figures 1 and 2, respectively. Often, as in the Needham-Schroeder specification, we omit the state events for brevity when they are not needed for our purposes.

In both protocols, each session is initiated with a spell to introduce the session-specific secrets and a corresponding cabal. In addition, the Otway-Rees protocol introduces a non-secret nonce N in rule 1.

Global State Transitions. Given a protocol P and a set of initial knowledge I (of the spy), the *global succession* relation transforms a state H to a new state H'. A succession is either *honest*, i.e. it corresponds to an action by an agent following the protocol, or it is *faked* by the spy.

- H' is an *honest* successor of H, denoted by $honest(P)(H, H')$, if there exists an applicable transition t in P such that $H' = (H \backslash (Pre(t) \cap States)) \cup Post(t)$.

- H' is a *fake* successor of H, denoted by $fake(I)(H, H')$, if there exists a field $X \in \mathsf{fake}(Cont(H) \cup I)$ such that $H' = H \cup \{X\}$.

In the honest case, a transition t is *applicable* in H if $Pre(t) \subseteq H$ and $New(t) \subseteq unused(H)$. In the fake case, the spy is restricted to adding only messages that can be inferred from the content of the current state and the initial knowledge. In either case, we write $global(P, I)(H, H')$. This relation determines a logical transition system with the empty set of events as its initial state. The set of reachable states of this transition system is denoted by $reachable(P, I)$.

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

Lemma 1 (Disjoint Book) *If $C, C' \in H \in reachable(P, I)$ then either $C = C'$ or $Book(C)$ and $Book(C')$ are disjoint.*

Secrecy Policy. A spell is *compatible* with an initial knowledge set I that does not mention its associated basic secrets.

$$compatible(I) \stackrel{\text{def}}{=} \{C \mid Sec(C) \cap \mathsf{parts}(I) = \emptyset\}$$

Given the spy's initial knowledge I, a global state H is called I-*discreet* if $Cont(H) \subseteq \mathcal{C}(Sec(C))$ for all I-compatible spells $C \in H$; these states are collected in the set $discreet(I)$. Now, a protocol P is called *discreet* if $discreet(I)$ is an invariant of the transition relation associated with P; i.e. for all I, $reachable(P, I)$ is a subset of $discreet(I)$.

Secrecy Theorem. As in [MR00], the Secrecy Theorem serves to split the secrecy proof for a protocol into a protocol-independent part and a protocol-dependent part. The protocol-dependent part is expressed by the occultness property. It says that if the prior state is discreet, the next message event generated by the protocol does not compromise a secret.

Some more notation needs to be introduced before defining occultness. A P-*configuration* is a tuple (I, H, C) such that $H \in reachable(P, I)$, $H \in discreet(I)$, $C \in compatible(I)$, and $C \in H$. Now, a protocol P is said to be *occult* if for all P-*configurations* (I, H, C) and for each applicable transition t in P,

$$Cont(Post(t)) \subseteq \mathcal{C}(Sec(C)).$$

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

Theorem 1 (Secrecy Theorem) *A protocol P is discreet iff it is occult.*

This theorem reduces secrecy proofs to proving occultness of individual rules of the protocol. In the case of the Otway-Rees protocol in Figure 1, for example, we are reduced to showing occultness of the rules (1),(2),(3), since occultness holds trivially for rule (0). (The fourth rule is also easy to handle.) For rule 1 of the Otway-Rees protocol we have to prove that for all reachable and I-discreet global states H, and for all spells C in H, compatible with I, it follows from the applicability conditions

- $\{N_a, \_\} \ddagger \{A, B\} \in H$,
- $\mathsf{A}_1(A, B, Srv) \in H$, and
- $N \in unused(H)$

that $[N, A, \{N_a, N, B\}_{\mathsf{shr}(A)}] \in \mathcal{C}(Sec(C))$. To establish this, we have to check two cases, depending on whether C is the spell in the rule or not. If it is, we note that $\mathsf{shr}(A)$ is in the coideal; in the other case, there is no secret to protect, because the Disjoint Book Lemma implies that N_a is not in $Book(C)$. This case split argument is one of the tasks that are simplified away using the search procedure we will present.

It is undecidable whether or not a given protocol P is occult. Undecidability of protocol security is well known, and has been proved in several different models. See, for example, [DLMS99] and its references. A proof for this particular model works by a simple encoding of the reachability problem of Turing machines such that the encoded Turing machine reaches its final state iff the protocol is not occult (see Appendix B). Using Theorem 1 it follows that it is also undecidable whether or not a given protocol is discreet.

$$\emptyset \xrightarrow{\{N_a, N_b\}} \{\{N_a, N_b\} \ddagger \{A, B\}\} \quad (0)$$

$$\{\{N_a, \_\} \ddagger \{A, B\}\} \xrightarrow{\emptyset} \{[\{N_a, A\}_{\mathsf{pub}(B)}]\} \quad (1)$$

$$\left\{ \begin{array}{l} \{N_b, \_\} \ddagger \{A, B\}, \\ [\{N_a, A\}_{\mathsf{pub}(B)}] \end{array} \right\} \xrightarrow{\emptyset} \{[\{N_a, N_b, B\}_{\mathsf{pub}(A)}]\} \quad (2)$$

$$\{[\{N_a, N_b, B\}_{\mathsf{pub}(A)}]\} \xrightarrow{\emptyset} \{[\{N_b\}_{\mathsf{pub}(B)}]\} \quad (3)$$

Figure 2. Encoding of Needham-Schroeder-Lowe protocol.

3 Branches

Occultness proofs work by case analysis. In proving occultness of rule 1 of the Otway-Rees protocol in Figure 1, for example, one tries to prove that $[N, A, \{N_a, N, B\}_{\mathsf{shr}(A)}] \in \mathcal{C}(Sec(C))$. Using the definition of ideals we are reduced to showing that $N \in \mathcal{C}(Sec(C))$, $A \in \mathcal{C}(Sec(C))$, and $\{N_a, N, B\}_{\mathsf{shr}(A)} \in \mathcal{C}(Sec(C))$ The second case is shown to be true immediately, since agent names are not elements of ideals. Furthermore, using the definition of ideals, the third case can be simplified further to the disjunction $\mathsf{shr}(A) \in Sec(C)$ or $[N_a, N, B] \in \mathcal{C}(Sec(C))$. In general, a field M is in the coideal generated by $Sec(C)$ iff for each nonce or key Z in $\mathsf{parts}(M)$, either Z is not in $Sec(C)$ or Z is encrypted with at least one key in $Sec(C)$.

This observation suggests that, instead of examining M itself, we examine the basic secrets occurring in it and the keys protecting them. A *branch* is a pair consisting of a basic field and a set of keys. The following recursion computes the branches occurring in a field M.

Definition 1 $\mathsf{bnch}(M)$ *is defined as* $\mathsf{bnch}(M, \emptyset)$*, where*

$$\begin{array}{rcl} \mathsf{bnch}(N, K_s) & = & \{(N, K_s)\} \\ \mathsf{bnch}(K, K_s) & = & \{(K, K_s)\} \\ \mathsf{bnch}(A, K_s) & = & \emptyset \\ \mathsf{bnch}([M_1, M_2], K_s) & = & \mathsf{bnch}(M_1, K_s) \cup \\ & & \quad \mathsf{bnch}(M_2, K_s) \\ \mathsf{bnch}(\{M\}_K, K_s) & = & \mathsf{bnch}(M, K_s \cup \{K\}) \end{array}$$

Thus,

$$\begin{array}{l} \mathsf{bnch}([N, A, \{N_a, N, B\}_{\mathsf{shr}(A)}]) = \\ \qquad \{(N, \emptyset),\ (N_a, \{\mathsf{shr}(A)\}),\ (N, \{\mathsf{shr}(A)\})\}. \end{array}$$

It turns out that a field M is in $\mathcal{C}(S)$ if and only if its branches satisfy a simple condition. The proof is by induction on the operator depth of M.

Proposition 1 *Let S be a set of basic fields; then:*

$M \in \mathcal{C}(S)$ iff for all $(Y, K_s) \in \mathsf{bnch}(M)$:
$Y \notin S$ or $K_s^{-1} \cap S \neq \emptyset$.

Definition 2 *For a protocol P, a branch $b = (Y, K_s)$, E_s a set of events, and N_s a set of nonces, the predicate $occ(P, b)(E_s, N_s)$ is defined to hold iff for all P-configurations (I, H, C) such that*

1. *$E_s \subseteq H$,*
2. *$N_s \subseteq unused(H)$, and*
3. *$Y \in Sec(C)$*

it is the case that $K_s^{-1} \cap Sec(C) \neq \emptyset$. For a transition t we write $occ(P, b)(t)$ instead of $occ(P, b)(Pre(t), New(t))$.

The following characterization of protocol occultness is a straightforward consequence of Proposition 1.

Proposition 2 *A protocol P is occult iff*

1. *for all transitions $t \in P$,*
2. *for all message fields M such that $M \in Post(t)$, and*
3. *for all branches $b \in \mathsf{bnch}(M)$*

the predicate $occ(P, b)(t)$ holds.

4 A Search Procedure for Establishing Occultness

Now, we describe a search procedure for establishing $occ(P, b)(t)$ for a given branch b and a transition t. This algorithm proceeds by applying some *basic tests* which are sufficient for establishing that the occultness predicate above holds. Whenever these tests fail, a *back step* is performed. Such a step explores every possibility of how certain message fields could have been published on the network.

Lemma 2 (Basic Tests) *Let $b = (Z, K_s)$ be a branch, E_s a set of events, and N_s a set of nonces; then: $occ(P, b)(E_s, N_s)$ holds if one of the following is true.*

1. *$Z \in N_s$*
2. *There exists a K'_s such that $K'_s \subseteq K_s$ and $(Z, K'_s) \in \mathsf{bnch}(Cont(E_s))$; in this case we write $(Z, K_s) \mathrel{\tilde{\in}} \mathsf{bnch}(Cont(E_s))$.*
3. *There exists a spell $C \in E_s$ such that $Z \in Book(C)$ and $K_s^{-1} \cap Sec(C) \neq \emptyset$; in this case we write $db(E_s, Z, K_s)$.*

Note that we employ the obvious extension of $\mathsf{bnch}(.)$ to sets of fields. The operator $\tilde{\in}$ says that a branch may have more keys than necessary, which is not harmful, since one good key is enough.

Given a P-configuration (I, H, C) such that the requirements listed in Definition 2 hold, Lemma 2 is proved as follows. First, consider the basic test $Z \in N_s$. Since $N_s \subseteq unused(H)$, it follows that $Z \notin Sec(C)$. Thus, $occ(P, b)(E_s, N_s)$ holds. Second, assume $(Z, K_s) \mathrel{\tilde{\in}} \mathsf{bnch}(Cont(E_s))$ and let $K'_s \subseteq K_s$ be such that $(Z, K'_s) \in \mathsf{bnch}(Cont(E_s))$. Since $E_s \subseteq H$ and H is I-$discreet$, it follows that $Cont(E_s) \subseteq \mathcal{C}(Sec(C))$. Consequently, using Lemma 1, ${K'_s}^{-1} \cap Sec(C) \neq \emptyset$, and thus ${K_s}^{-1} \cap Sec(C) \neq \emptyset$. The third part of Lemma 2 is a consequence of the disjoint book lemma (Lemma 1).

Consider, for example, rule 2 of the Otway-Rees protocol in Figure 1. This rule, denoted by $or2$, contains a message variable X in its pre. Thus, $or2$ denotes an infinite set of transitions, and a uniform proof of the occultness of this family of transitions starts by introducing a symbolic constant X'.

$$\begin{array}{l} (N, \emptyset), (N, \{\mathsf{shr}(B)\}) \mathrel{\tilde{\in}} \\ \qquad \mathsf{bnch}(Cont(\{[N, A, X'],\ \{N_b, \_\} \ddagger \{A, B\}\}))\ , \end{array}$$

it follows that both

$$occ(OR, (N, \emptyset))(or2)$$

and

$$occ(OR, (N, \{\mathsf{shr}(B)\}))(or2)$$

hold. Furthermore, since the predicate $db(\{[N, A, X'],\ \{N_b, \_\} \ddagger \{A, B\}\}, N_b, \{\mathsf{shr}(B)\})$ holds, it follows that $occ(OR, (N_b, \{\mathsf{shr}(B)\}))(or2)$ holds, too.

The occultness proof of the Otway-Rees protocol uses only basic tests. In general, however, other rules have to be taken into consideration. Consider, for example, the case $(N_a, \{\mathsf{pub}(A)\}) \in \mathsf{bnch}(\{N_a, N_b, B\}_{\mathsf{pub}(A)})$ for proving rule 2 of the Needham-Schroeder-Lowe protocol in Figure 2; this rule is denoted by $nsl2$. None of the basic tests above establishes that $occ(NSL, (N_a, \{\mathsf{pub}(A)\}))(nsl2)$ holds. The purpose of a *back step* is to obtain additional information for applying the basic tests. For each message event M in E_s, two possibilities have to be taken into consideration: either M has been published by an honest agent following the protocol rules or M was injected by the intruder.

Definition 3 (Search) *Let P be a protocol, t be a transition of P, and b be a branch of the form (Z, K_s); then:*

$$main(P,b)(t) \stackrel{def}{=} Z \in New(t) \vee search(P,b)(Pre(t))$$

$$search(P,b)(E_s) \stackrel{def}{=} b \,\tilde{\in}\, \mathsf{bnch}(Cont(E_s)) \ \vee \ db(E_s, Z, K_s) \ \vee \ back(P,b)(E_s)$$

$$back(P,b)(E_s) \stackrel{def}{=} (\exists M \in E_s :\ Z \in \mathsf{parts}(M)) \ (honest(P,b)(M) \wedge fake(P,b)(M))$$

$$honest(P,b)(M) \stackrel{def}{=} (\forall t' \in P, M \in \mathsf{parts}(Cont(Post(t')))) \ (M \in \mathsf{parts}(Cont(Pre(t')))) \vee search(P,b)(Pre(t'))$$

$$fake(P,b)(M) \stackrel{def}{=} (\forall M_1,\ldots,M_n : [M_1,\ldots,M_n] = M) \ \vee_{M_i}(Z \in \mathsf{parts}(M_i) \wedge search(P,b)(\{M_i\}))$$

These predicates determine a search procedure in the usual way. For example, $P_1 \vee P_2$ is computed nondeterministically: if the computation of P_1 (or P_2) terminates with $true$, then the computation of $P_1 \vee P_2$ terminates with $true$. Using these conventions, Definition 3 gives rise to a nondeterministic proof procedure for establishing occultness.

Now we outline the proof of soundness for our procedure. The proof of the main lemma applies induction on the number of back steps in deducing that predicate $search(P, Z, K_s)(E_s)$ holds. A detailed proof can be found in the appendix.

Lemma 3 (Main Lemma) *Let P be a protocol, b be some branch, and E_s a set of of events; then:*

If the predicate $search(P,b)(E_s)$ holds, then $occ(P,b)(E_s, \_)$ holds, too.

Altogether, soundness of the search procedure follows from the Lemmas 2 and 3, and the secrecy theorem (Theorem 1).

Theorem 2 (Soundness) *Let P be a protocol. If $main(P,b)(t)$ holds*

- *for all transitions $t \in P$,*
- *for all message events $M \in Post(t)$, and*
- *for all branches $b \in \mathsf{bnch}(M)$,*

then P is discreet.

Our method, however, is not complete. If one of the proof obligations can not be shown to hold, then one may not necessarily conclude that the protocol is not discreet. Moreover, there are occult protocols for which our search procedure does not terminate; such an example can be found in Section 5.

Let us return to proving occultness of the rule $nsl2$; for the branch $b = (N_a, \{\mathsf{pub}(A)\})$ the derivation starts as follows.

$$\begin{aligned}
& main(NSL,b)(nsl2) \\
\iff \ & search(NSL)(b)(\{\{N_b|_\} \ddagger \{A,B\}, [\{N_a, A\}_{\mathsf{pub}(B)}]\}) \\
\iff \ & back(NSL)(b)(\{\{N_b|_\} \ddagger \{A,B\}, [\{N_a, A\}_{\mathsf{pub}(B)}]\}) \\
\iff \ & honest(NSL,b)(\{N_a, A\}_{\mathsf{pub}(B)}) \ \wedge \ fake(NSL,b)(\{N_a, A\}_{\mathsf{pub}(B)})
\end{aligned}$$

Since only the first rule of the Needham-Schroeder-Lowe protocol contains a message of the form $\{N_a, A\}_{\mathsf{pub}(B)}$ in its *post*,

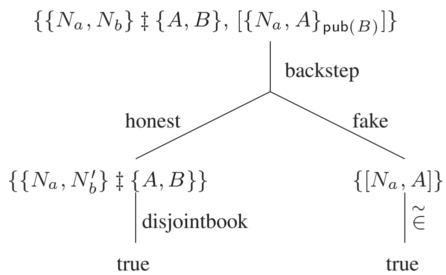
$$\begin{aligned}
& honest(NSL,b)(\{N_a, A\}_{\mathsf{pub}(B)}) \\
\iff \ & search(NSL,b)(\{\{N_a|_\} \ddagger \{A,B\}\}) \\
\iff \ & true
\end{aligned}$$

This reduces to $true$ because of the disjoint book test. Furthermore,

$$\begin{aligned}
& fake(NSL,b)(\{N_a, A\}_{\mathsf{pub}(B)}) \\
\iff \ & search(NSL,b)(N_a, A) \\
\iff \ & true
\end{aligned}$$

since $(N_a, \{\mathsf{pub}(A)\}) \,\tilde{\in}\, \mathsf{bnch}([N_a, A])$. Consequently, rule $nsl2$ is occult.

Derivations based on the predicates in Definition 3 can be visualized as search trees. These search trees have set of events as nodes, the edges are labeled either with a basic test or with the name of one of the search steps. A leaf is $true$ if one of the basic tests succeeds, and $false$ if all the basic tests fail and if there are no more messages in the set of events of the parent node. Branching corresponds to a conjunction, and disjunctions are realized by copying derivation trees. For the rule $nsl2$ and the branch $(N_a, \{\mathsf{pub}(A)\})$, for example, the run of $search$ is visualized as follows.



In general, the search tree generated by the predicates in Definition 3 may be infinitely branching whenever there is an infinite set of protocol transitions. However, the set of honest transitions is usually generated by a finite set of rules of the form $rl = Pre(rl) \longrightarrow Post(rl)$, such that each transition t of the protocol is obtained by a substitution σ, i.e. $Pre(t) = Pre(rl)\sigma$, $New(t) = New(rl)\sigma$, etc. The remainder of this section is devoted to lifting the results above from transitions to rules. In this way, we obtain occultness proof obligations for rules which possibly contain variables.

The notion of branches has to be extended to include message fields containing variables X by adding the case $\mathsf{bnch}(X, K_s) = \{(X, K_s)\}$ to Definition 1. Now, the search algorithm in Definition 3 is lifted to this new case of field variables in branches.

Definition 4 *Let P be a protocol, b be a branch, and rl be a rule; then:*

$$main'(P,b)(rl) \overset{def}{=} \begin{cases} b \mathrel{\tilde{\in}} \mathsf{bnch}(Cont(Pre(rl))) \\ \quad \text{if } b = (Z, \_) \text{ and } Z \text{ is a variable;} \\ main(P,b)(rl) \\ \quad \text{otherwise.} \end{cases}$$

The soundness of this extension follows from the following fact.

Lemma 4 *If $main'(P,b)(Pre(rl), New(rl))$ holds for all $M \in Post(rl)$, for all $b \in \mathsf{bnch}(M)$, then*

- *for all instances t of rule rl,*
- *for all message events $M' \in Post(t)$,*
- *for all branches $b \in \mathsf{bnch}(M')$*

the predicate $main(P,b)(Pre(t), New(t))$ holds.

Let $b \in \mathsf{bnch}(M')$ such that $M' \in Post(t)$. If $b \in \mathsf{bnch}(M)$, then the predicate $main(P,b)(Pre(t), New(t))$ holds by the definition of $main'(P,b)(Pre(rl), New(rl))$. Otherwise, if $b = (Z, K_s)$ comes from an instantiation σ of a field variable, there exists $X \in \mathsf{parts}(M)$ such that $(X, K_1) \in \mathsf{bnch}(M)$ and $(Z, K_2) \in \mathsf{bnch}(X\sigma)$ with $K_s = K_1 \cup K_2$. Now, $main'(P, (X, K_1))(Pre(rl), New(rl))$ holds, and consequently $(X, K_1) \mathrel{\tilde{\in}} \mathsf{bnch}(Cont(Pre(rl)))$, $b \mathrel{\tilde{\in}} \mathsf{bnch}(Cont(Pre(rl\sigma)))$, and finally $main(P,b)(Pre(t), New(t))$ hold. This finishes the proof of Lemma 4.

Theorem 3 *Let P be a protocol. If $main'(P,b)(Pre(rl), New(rl))$ holds*

- *for all rules $rl \in P$,*
- *for all message events $M \in Post(rl)$, and*
- *for all branches $b \in \mathsf{bnch}(M)$,*

then P is discreet.

$$\begin{array}{rcll}
\emptyset & \xrightarrow{\{N_b, K_{ab}\}} & \{\{N_b, K_{ab}\} \ddagger \{A, B\}\} & (0) \\
\emptyset & \xrightarrow{N_a} & \{[A, N_a]\} & (1) \\
\left\{\begin{array}{l}\{N_b, K_{ab}\} \ddagger \{A, B\}, \\ {[A, N_a]}\end{array}\right\} & \xrightarrow{\emptyset} & \{[B, \{A, N_a, N_b\}_{\mathsf{shr}(B)}]\} & (2) \\
\left\{\begin{array}{l}\{N_b, K_{ab}\} \ddagger \{A, B\}, \\ {[B, \{A, N_a, N_b\}_{\mathsf{shr}(B)}]}\end{array}\right\} & \xrightarrow{\emptyset} & \{[\{B, K_{ab}, N_a, N_b\}_{\mathsf{shr}(A)}, & (3) \\
 & & \quad \{A, K_{ab}\}_{\mathsf{shr}(B)}]\} & \\
\{[\{B, K_{ab}, N_a, N_b\}_{\mathsf{shr}(A)}, X\}] & \xrightarrow{\emptyset} & \{[X, \{N_b\}_{K_{ab}}]\} & (4)
\end{array}$$

Figure 3. Encoding of the Yahalom protocol.

$$\begin{array}{rcll}
\emptyset & \xrightarrow{\{N_a\}} & \{\{N_a\} \ddagger \{A_1, A_2\}\} & (0) \\
\{\{N_a\} \ddagger \{A_1, A_2\}\} & \xrightarrow{\emptyset} & \begin{array}{l}\{[A_1, A_2, \cdots, A_n, \\ \{A_1, N_a\}_{\mathsf{shr}(A)}]\}\end{array} & (1) \\
\begin{array}{l}\{[A_1, A_2, \cdots, A_n, \\ \quad \{A_1, N_a\}_{\mathsf{shr}(A)}]\}\end{array} & \xrightarrow{\emptyset} & \{[\{A_1, A_2, N_a\}_{\mathsf{shr}(A_1)}]\} & (2)
\end{array}$$

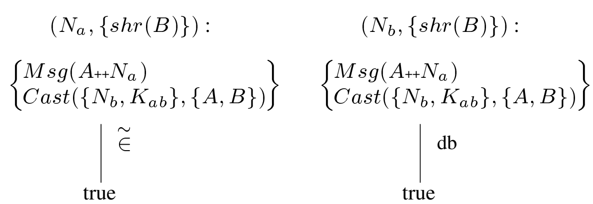
Figure 4. A protocol which requires at least n back steps for proving occultness.

5 Examples

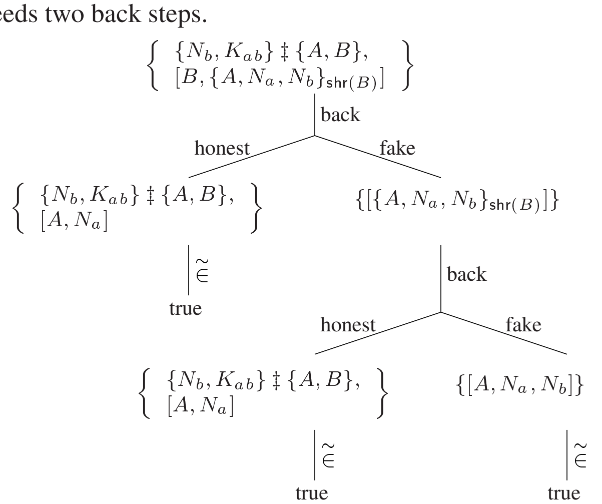
In the previous sections, we have already demonstrated that the Otway-Rees protocol can be proved to be occult using only basic tests. Likewise, the occultness proof of the

Needham-Schroeder-Lowe protocol requires at most one back step for each rule and each branch. Here we give an overview of the proof of Yahalom's protocol, which requires up to three back steps for proving occultness. Moreover, we demonstrate the incompleteness of our algorithm with an example of an occult protocol for which the search procedure is non-terminating. Then, we give an example of a protocol that requires at least n back steps in proving occultness. Finally, we use a failed proof attempt of the original Needham-Schroeder protocol to show that it is indeed not occult.

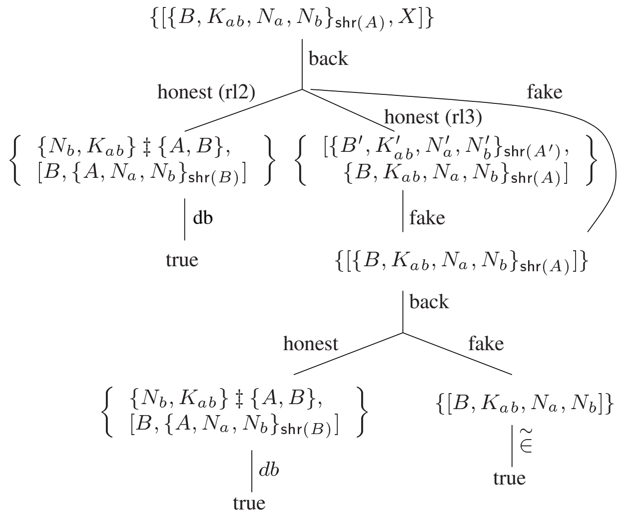
Yahalom Protocol. This protocol has been studied extensively by Paulson [Pau97]. An encoding of the Yahalom protocol (without state events) can be found in Figure 3. Occultness of the initial rule (1) is obvious. For verifying occultness of rule (2) we have to consider the two branches $(N_a, \{\mathsf{shr}(B)\})$, $(N_b, \{\mathsf{shr}(B)\})$ of the single message in the post.



In verifying occultness of rule (3) four branches have to be considered. Occultness for the cases $(N_b, \{\mathsf{shr}(A)\})$, $(K_{ab}, \{\mathsf{shr}(A)\})$, and $(K_{ab}, \{\mathsf{shr}(B)\})$ is established using the disjoint book test, whereas the branch $(N_a, \{\mathsf{shr}(A)\})$ needs two back steps.

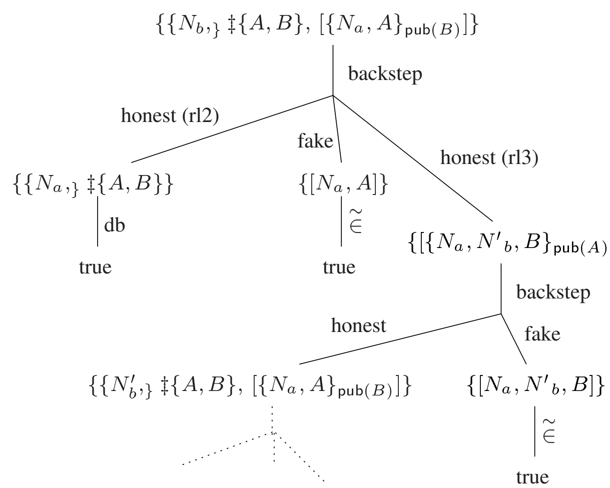


Finally, the branches $(X, \emptyset)$ and $(N_b, \{K_{ab}\})$ have to be considered for establishing occultness of the rule (4). The proof for the $(X, \emptyset)$ branch only needs the basic test "$\tilde{\in}$ $\mathsf{bnch}(\ldots)$" and the following proof for the branch $(N_b, \{K_{ab}\})$ requires three back steps.

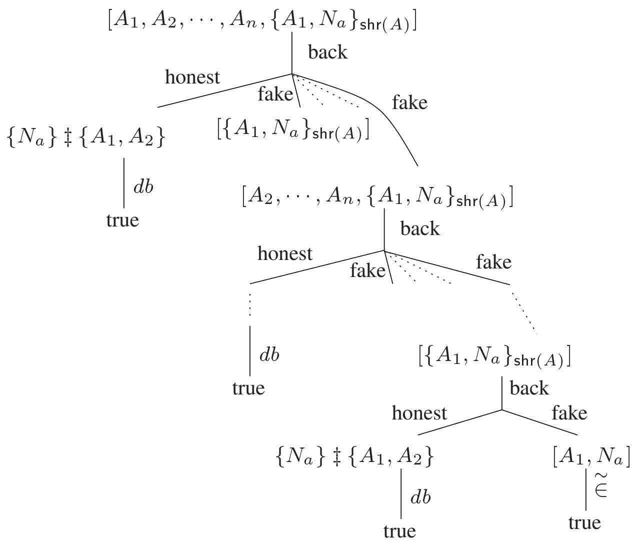


Altogether, the Yahalom protocol is occult.

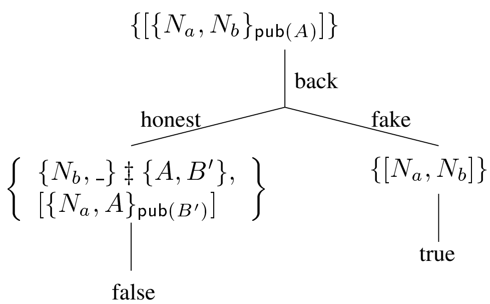
Non terminating procedure. We slightly modify the Needham-Schroeder-Lowe protocol : the encoding of this protocol is identical to the one in Figure 2 except for the post of rule 3. This post is now assumed to be given by $\{[\{N_b\}_{\mathsf{pub}(B)}, \{N_a, A\}_{\mathsf{pub}(B)}]\}$. The protocol remains occult (the agent A just sends again the first message $\{N_a, A\}_{\mathsf{pub}(B)}$)) but for the rule 2 and the branch $(N_a, \{\mathsf{pub}(A)\})$, our procedure creates an infinite tree as visualized below. Consequently, our procedure fails to detect occultness of this protocol.



Arbitrary Number of Backsteps. Occultness of the Otway-Rees protocol is proved using only basic tests, the proof of the Needham-Schroeder-Lowe protocol needs at most one back step for verifying each occultness obligation, and the Yahalom protocol is proved using at most two back steps. In general, given a natural number n, there is an occult protocol which requires at least n back steps for proving occultness. Such a family of protocols is given in Figure 4. The proof tree for demonstrating occultness of the rule 2 of this protocol is given as follows; obviously, there is no deduction requiring fewer back steps.



Failed Proof Attempts. Lowe [Low96] showed that the original description of the Needham-Schroeder [NS78] protocol was flawed. The encoding of this protocol is identical to the one in Figure 2 except for the post of rule 2. This post is now assumed to be given by $\{[\{N_a, N_b\}_{\mathsf{pub}(A)}]\}$. Our search procedure terminates with an incomplete proof for this modified rule.



Using this failed proof attempt, we can show that the protocol is indeed not occult. The construction starts at the leaf labelled with $false$. Its parent node contains a cast and is exactly the Pre of one of the rules, say rl, of the protocol. Now, we consider a (partial) run of the protocol where all the rules preceding rl in the protocol description are applied in the given order.

$$\begin{array}{rcl} \emptyset & \xrightarrow{\{N_a, N_b\}} & \{\{N_a, N_b\} \ddagger \{A, B'\}\} \\ \{\{N_a, N_b\} \ddagger \{A, B'\}\} & \xrightarrow{\emptyset} & \{[\{N_a, A\}_{\mathsf{pub}(B')}]\} \end{array}$$

Next, we simulate an attack by following the branch from the $false$ leaf up to its root. The parent node of the $false$ leaf is directly connected with the root by an honest edge.

$$\left\{ \begin{array}{l} \{N_b, \_\} \ddagger \{A, B\}, \\ [\{N_a, A\}_{\mathsf{pub}(B')}] \end{array} \right\} \xrightarrow{\emptyset} \{[\{N_a, N_b\}_{\mathsf{pub}(A)}]\}$$

Having reached the root of the tree, one applies the rule for which our algorithm fails.

$$\{[\{N_a, N_b\}_{\mathsf{pub}(A)}]\} \xrightarrow{\emptyset} \{[\{N_b\}_{\mathsf{pub}(B)}]\}$$

Thus $\{N_b\}_{\mathsf{pub}(B)} \notin \mathcal{C}(Sec(\{N_a, N_b\} \ddagger \{A, B'\}))$, and the protocol is not occult.

6 Discussion

We have developed an automatic search procedure for proving the occultness of protocol rules and proved its correctness. If the procedure terminates with $true$, then the argument rule is occult. Moreover, occultness of all rules implies that the protocol is indeed secure. Our procedure follows the informal reasoning steps in [MR00], mechanizations do not require any user intervention, and there is a visually appealing graphical representation of occultness proofs.

We have tested our proof procedure on selected protocols from the the Clark and Jacob survey [CJ97]. Usually, we can prove occultness using only a small number of search space extensions. The Otway-Rees and the Carlson protocol, for example, are proved to be secure using only basic tests, the Needham-Schroeder protocol needs at most one back step for verifying occultness of each rule and branch, and the Yahalom protocol needs at most three back steps for verifying each occultness conditions. We have also given examples of protocols whose occultness proofs need at least n back steps for an arbitrary natural number.

Much work remains to be done. In order to deal with many protocols used in practice, we have to extend our methods and support protocol features like hashing and timestamps. The algorithm described here is not a semi-decision procedure in the sense that occultness is eventually detected. It may be interesting to investigate subclasses of protocols which only require a bounded number of back steps, and for which our algorithm acts as a decision procedure. Also, we do not yet know under what circumstances a failed proof attempt implies that the protocol is insecure. An advantage of our method seems to be that it permits constructing attacks from failed proof attempts. For example from the failed proof attempt for the original Needham-Schroeder protocol in Section 5 we can construct Lowe's man-in-the-middle attack. We plan to investigate methods for constructing such attacks from failed proof attempts.

References

[CDL+99] I. Cervesato, N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. A meta-notation for protocol analysis. In *12th IEEE Computer Security Foundations Workshop*, pages 55–69. IEEE Computer Society, 1999.

[CJ97] J. Clark and J. Jacob. A survey of authentication protocol literature. `http://www.cs.york.ac.uk/~jac/papers/drareviewps.ps`, 1997.

[Coh00] E. Cohen. TAPS: A first-order verifier for cryptographic protocols. In *13th IEEE Computer Security Foundations Workshop*, pages 144–158. IEEE Computer Society, 2000.

[DLMS99] N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. In *Formal Methods and Security Protocols*, Federated Logic Conference, 1999.

[DS97] B. Dutertre and S. Schneider. Using a PVS embedding of CSP to verify authentication protocols. In *Theorem Proving in Higher Order Logics, TPHOL's 97*, volume 1275 of *Lecture Notes in Computer Science*, pages 121–136. Springer-Verlag, August 1997.

[DSS01] B. Dutertre, H. Saïdi, and V. Stavridou. Intrusion-Tolerant Group Management in Enclaves. Accepted for publication at the International Conference on Dependable Systems and Networks (DSN'2001), 2001.

[Gon97] L. Gong. Enclaves: Enabling Secure Collaboration over the Internet. *IEEE Journal of Selected Areas in Communications*, 15(3):567–575, April 1997.

[Low96] 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.

[Mea00] C. Meadows. Invariant generation techniques in cryptographic protocol analysis. In *13th IEEE Computer Security Foundations Workshop*, pages 159–167. IEEE Computer Society, 2000.

[Mon99] D. Monniaux. Decision procedures for the analysis of cryptographic protocols by logics of belief. In *12th Computer Security Foundations Workshop*, Mordano, Italy, June 1999. IEEE Computer Society.

[MR00] J. Millen and H. Rueß. Protocol-independent secrecy. In *2000 IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2000.

[NS78] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–998, December 1978.

[OR87] D. Otway and O. Rees. Efficient and timely mutual authentication. *ACM Operating System Review*, 21(1):8–10, 1987.

[Pau97] L. Paulson. Relations between secrets: Two formal analyses of the Yahalom protocol. Technical Report TR432, University of Cambridge, Computer Laboratory, July 1997.

[Pau98] L. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1):85–128, 1998.

[RM00] H. Rueß and J. Millen. Local secrecy for stated-based models. In *Proc. of the Workshop on Formal Methods in Computer Security (FMCS'2000)*, Chicago, IL, 2000.

[Sch98] S. Schneider. Verifying authentication protocols in CSP. *IEEE Transactions on Software Engineering*, 24(9):741–758, September 1998.

[THG98] 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.

A Proof of the Main Lemma

Main Lemma:

If the predicate $search(P,(Z,K_s))(E_s)$ holds, then $occ(P,Z,K_s)(E_s,-)$ holds, too.

Proof : Instead of proving that $occ(P,Z,K_s)(E_s,N_s)$ holds, we prove the stronger property $occstrong(P,Z,K_s)(E_s)$ defined to hold iff for all P-configurations (I,H,C) such that

1. $Nonstates(E_s) \subseteq H$ and
2. $Z \in Sec(C)$

it is the case that $S \cap Sec(C) \neq \emptyset$,. The set $Nonstates(E_s)$ includes all non-state events in E_s. Obviously, $occstrong(P,Z,K_s)(E_s)$ implies $occ(P,Z,K_s)(E_s,-)$. The proof is by induction on the minimum number of back steps for deriving that $search(P,Z,K_s)(E_s)$ holds.

Initialization. If the derivation of $search(P, Z, K_s)(E_s)$ terminates with *true* and if no back steps has been used, then either $(Z, K_s) \ \widetilde{\in} \ \mathsf{bnch}(Cont(E_s))$ or $db(E_s, Z, K_s)$ holds. Using basic tests one concludes that $occstrong(P, Z, K_s)(E_s)$ holds in both cases.

Step. Assume that $occstrong(P, Z, K_s)(E_s)$ holds for every $search(P, Z, K_s)(E_s)$ with a derivation that uses n or fewer back steps. Furthermore, consider P, Z, K_s, and E_s such that the derivation of $search(P, Z, K_s)(E_s)$ terminates with $true$ and uses $n+1$ back steps, and assume a P-configuration (I, H, C) such that $Nonstates(E_s) \subseteq H$ and $Z \in Sec(C)$.

Consequently, the back step terminates with $true$, and there exists a $M \in E_s$ and $Z \in \mathsf{parts}(M)$, such that $honest(P, Z, K_s)(M) \wedge fake(P, Z, K_s)(M) = true$ and the derivation of $honest(P, Z, K_s)(M)$ and $fake(P, Z, K_s)(M)$ uses at most n back steps. Now, M is in E_s, so M is in H. By induction on H there exist two global states H_1, H_2 such that

$$global(P, I)(H_1, H_2) \,\&\, Nonstates(H_1) \subseteq H \,\& \\ M \in \mathsf{parts}(Cont(H_2)) \,\&\, M \notin \mathsf{parts}(Cont(H_1)).$$

Apply case analysis depending on whether the global extension is honest or faked.

Case $honest(P)(H_1, H_2)$**:** There exists an applicable transition $t \in P$ such that $H_2 = Post(t) \cup (H_1 \backslash (Pre(t) \cap States))$; thus, $Nonstates(Pre(t)) \subseteq H$ and $M \in \mathsf{parts}(Cont(Post(t)))$. Since $honest(P, Z, K_s)(M)$ reduces to $true$ and $M \in \mathsf{parts}(Cont(Post(t)))$, we have two cases : either $M \in \mathsf{parts}(Cont(Pre(t)))$ or $search(P, Z, K_s)(Pre(t))$ holds. In the first case, we obtain a contradiction immediately: $M \in \mathsf{parts}(Cont(Pre(t)))$ and $Pre(t) \in H_1$ implies $M \in \mathsf{parts}(Cont(H_1))$. Thus, $search(P, Z, K_s)(Pre(t))$ holds, and its derivation uses at most n back steps. Thus, $occstrong(P, Z, K_s)(Pre(t))$ holds. Because of the facts $Nonstates(Pre(t)) \in H$ and $Z \in Sec(C)$, it follows that $S \cap Sec(C) = \emptyset$. Consequently, the predicate $occstrong(P, Z, K_s)(E_s)$ holds.

Case $fake(I)(H_1, H_2)$**:** By definition of $fake$, $H_2 = H_1 \cup \{M'\}$ where $M' \in \mathsf{fake}(Cont(H_1) \cup I)$. Since $M \in \mathsf{parts}(Cont(H_2))$ and $M \notin \mathsf{parts}(Cont(H_1))$, we know that $M \in \mathsf{parts}(\mathsf{fake}(Cont(H_1) \cup I))$. It is easy to verify that

$$\mathsf{parts}(\mathsf{fake}(Cont(H) \cup I)) = \\ \mathsf{fake}(Cont(H) \cup I) \cup \mathsf{parts}(Cont(H) \cup I) \ .$$

In addition, $M \notin \mathsf{parts}(I)$ (unless $Z \in \mathsf{parts}(I)$, in which case $Z \notin Book(C)$ by choice of C, which contradicts the hypothesis $Z \in Book(C)$) and $M \notin \mathsf{parts}(Cont(H_1))$, thus $M \notin \mathsf{parts}(Cont(H) \cup I)$. Consequently, M must have been synthesized, meaning $X \in \mathsf{fake}(Cont(H_1) \cup I)$ if $M = \{X\}_K$ or there exist M_1, M_2 such that $M = M_1, M_2$ and $M_1, M_2 \in \mathsf{fake}(Cont(H_1) \cup I) \subset \mathsf{fake}(Cont(H) \cup I)$.

Now, let $H' = H \cup \{M_1, M_2\}$ (respectively $H' = H \cup \{X\}$). It is easy to verify that (I, H', C) is still a P-configuration, and we get $\{M_1, M_2\} \in H'$ (respectively $\{X\} \in H'$) and $Z \in Sec(C)$.

Assume (without loss of generality) that $Z \in \mathsf{parts}(M_1)$. By definition of $fake(I)(H_1, H_2)$, $search(P, Z, K_s)(\{M_1\})$ (respectively $search(P, Z, K_s)(\{X\})$) holds and its derivation uses at most n back steps. Therefore, the predicate $occstrong(P, Z, K_s)(\{M_1\})$ (resp. $occstrong(P, Z, K_s)(\{X\})$) holds. Finally, one concludes that $occstrong(P, Z, K_s)(E_s)$ holds.

■

B Proof of undecidability of occultness

Undecidability:

It is undecidable wether or not a given protocol P is occult.

Proof sketch: We encode the reachability problem of Turing machines in such a way that encoded Turing machine reaches its final state iff the protocol is not occult.

Let T be a Turing machine, Q the set of states (q_0 is the initial state and q_f is the final state), Σ the tape alphabet ($\sharp$ is the blank symbol), its transitions are of the form $q_1\, a_1 \rightarrow q_2\, a_2, D$, where $q_1, q_2 \in Q$, $a_1, a_2 \in \Sigma$ and $D \in \{L, R, S\}$. The interpretation is : if the machine T is in state q_1 and its head points to a_1 then T changes to state q_2, replaces a_1 with a_2 and moves the head right (if $D = R$), left (if $D = L$) or remains in the current cell (if $D = S$).

Let $Agent$ be a set of agent constants such that

$$\Sigma \cup Q \subset Agent$$

and such that another special symbol $h \in Agent$.

The location of the head is designated by placing an h to the right of the symbol under the head.

We encode the transitions $q_1\, a_1 \rightarrow q_2\, a_2, R$ by the rules:

$$\big\{[\{K, q_1, X, a_1, h, b, Y\}_{\mathsf{shr}(A)}]\big\} \xrightarrow{\emptyset} \big\{[\{K, q_2, X, a_2, b, h, Y\}_{\mathsf{shr}(A)}]\big\} \quad \text{for all } b \in \Sigma$$

$$\big\{[\{K, q_1, X, a_1, h\}_{\mathsf{shr}(A)}]\big\} \xrightarrow{\emptyset} \big\{[\{K, q_2, X, a_2, \sharp, h\}_{\mathsf{shr}(A)}]\big\}$$

and the same rules where respectively X, X and Y, and Y are omitted.

L and S transitions are encoded in the same way.
Each message is of the form $[\{K,q,X\}_{\mathsf{shr}(A)}]$ where q holds for the current state and X describes the current tape.

$\mathsf{shr}(A)$ is a private key shared between the server and A but it could be any shared key between two agents.

The initialization rule is

$$\emptyset \quad \xrightarrow{\{K,N\}} \quad \big\{K \ddagger A, [\{K, q_0, \sharp, h\}_{\mathsf{shr}(A)}]\big\}$$

and the "final" rule is

$$\big\{[\{K, q_f, X\}_{\mathsf{shr}(A)}]\big\} \quad \xrightarrow{\emptyset} \quad \{[K]\}$$

The final state of the Turing machine is reachable iff this protocol is not occult. Indeed, K is revealed iff a message of the form $[\{K, q_f, X\}_{\mathsf{shr}(A)}$ is sent (where q_f is the final state of the Turing machine) which corresponds to the fact that q_f is reached.

■