

C3PO: a Tool for Automatic Sound Cryptographic Protocol Analysis

Anthony H. Dekker
Defence Science and Technology Organisation and
Department of Computer Science, Australian National University
Postal: PO Box 3925, Manuka ACT 2603, Australia
E-mail: dekker@acm.org

Abstract

In this paper we present an improved logic for analysing authentication properties of cryptographic protocols, based on the SVO logic of Syverson and van Oorschot. Such logics are useful in electronic commerce, among other areas. We have constructed this logic in order to simplify automation, and we describe an implementation using the Isabelle theorem-proving system, and a GUI tool based on this implementation. The tool is typically operated by opening a list of propositions intended to be true, and clicking one button. Since the rules form a clean framework, the logic is easily extensible. We also present in detail a proof of soundness, using Kripke possible-worlds semantics.

1. Introduction

During the past decade, there has been extensive interest in analysing the correctness of cryptographic protocols. Current interest in electronic commerce has added to this interest.

One set of solutions for analysing cryptographic protocols has been the BAN family of logics, beginning with the work of Burrows, Abadi, and Needham [1, 2] and continuing with the GNY logic of Gong, Needham, and Yahalom [3, 4]. These logics allow reasoning about authentication, but their complexity has resulted in problems with unsoundness and difficulty of implementation. In spite of the published soundness proof of BAN logic in [2], the BAN and GNY logics have been extensively criticised (e.g. [3, 4, 11]).

One of the more recent members of this family is the SVO logic of Syverson and van Oorschot [11]. The SVO logic unifies much of the previous work in an elegant framework, and is proved to be sound (which we view as essential, since we are particularly interested in the information security evaluation of cryptographic

protocols at a high level). However, this logic is not suitable for automation with theorem provers such as Isabelle. Such automation is of great value for practical analysis of cryptographic protocols, and previous work has been done on automating the GNY logic (e.g. [5, 6, 7, 8, 9, 10]) which will discuss later in section 8.

In this paper we present an improvement to this logic which we call SVD logic. We have designed this logic with two things in mind: first, ease of implementation in the Isabelle theorem-proving system. This has required developing terminating proof tactics, and addressing all possible cases of circularity in the rules. We have demonstrated the success of our tactics by building a GUI tool which automates the proof process. Since the rules have been designed with automation in mind, the proof tactics required are fairly simple, and automation in a system other than Isabelle should be relatively easy. Because the set of rules forms a clean framework, it should also be easily extensible. Our tool is intended for users with no training in Isabelle, and is typically operated by opening a list of propositions intended to be true, and clicking one button.

Achieving an easily implemented set of rules has resulted in a relatively large number of rules, and so our second concern has been the soundness of the rules. We have achieved soundness by keeping the rules very simple (more complex rules have been derived as theorems) and giving the rules a more operational flavour, by casting them in terms of *operations* which a principal is capable of performing. We have developed a formal proof of soundness using Kripke possible-worlds semantics.

2. The Rules

We consider messages to be made up of *terms*, which are built out of atomic elements and the following operators:

$(x!y)$ Pairing

$\{x\}k$ Public- or symmetric-key encryption

$\{x\}^\circ k$ Symmetric-key decryption

There is an important reason why we consider terms rather than bit patterns: a theory based on bit patterns can result in undesirable behaviour. For example, an arbitrary bit pattern x could be viewed as $\{y\}k$ for some symmetric key k even when there is no reason to do so. There would then be no such thing as an atomic bit pattern, since this process could be continued ad infinitum. The fact that we can construct any arbitrary bit pattern also conflicts with the fact that any key is a bit pattern, but that we cannot construct every key. Using terms, we consider the term $\{y\}k$ to exist only when there was a term y which was actually encrypted with k by somebody.

We write $+k$ and $-k$ for the public and private keys associated with a public-key pair k . This gives us the following rules about encryption (where \equiv denotes equality of terms):

K1 $\{\{x\}k\}^\circ k \equiv x$ for symmetric keys k

K2 $\{\{x\}+k\}-k \equiv x$

K3 $\{\{x\}-k\}+k \equiv x$

Notice that the syntax $\{x\}k$ is ambiguous, but the type-checking facilities of Isabelle ensure that this does not cause any problems. The syntax \hat{k} provides a useful shorthand for the inverse of a key which can be expressed easily using typed variables in Isabelle:

K4 $+\hat{k} \equiv -k$

K5 $-\hat{k} \equiv +k$

K6 $\hat{\hat{k}} \equiv k$ for symmetric keys k

One-way hash functions are treated as keys for which no inverse exists. We use the syntax $k \mathbf{DH} l$ for the Diffie-Hellman key generated from the public key pairs k and l . The syntax $\mathcal{P} \mathbf{PK} +k$ means that $+k$ is the public key for the principal \mathcal{P} , where a principal is any party (whether a person or a software agent) involved in a cryptographic protocol. The syntax $\mathcal{P} \xleftrightarrow{k} Q$ means that k is a good symmetric key for the principals \mathcal{P} and Q (here \iff means ‘if and only if’):

K7 $k \mathbf{DH} l \equiv l \mathbf{DH} k$

K8 $\mathcal{P} \xleftrightarrow{k} Q \iff Q \xleftrightarrow{k} \mathcal{P}$

K9
$$\frac{\mathcal{P} \mathbf{PK} +k, \quad Q \mathbf{PK} +l}{\mathcal{P} \xleftrightarrow{k \mathbf{DH} l} Q}$$

The rule K9 says that a good symmetric key can be generated from two public key pairs (it corresponds to rule 5 in the SVO system, and K8 corresponds to rule 19). In this paper we use \mathbf{PK} for public keys, whether used for authentication, encryption, or key agreement; but these can easily be distinguished by using different symbols if desired.

The rules T1–T4 define the obvious subterm relation on terms. We refer to this as the *strict* subterm relation, in contrast to the *loose* subterm relation, which will define later:

T1 $\vdash x \subseteq x$

T2
$$\frac{z \subseteq x}{z \subseteq (x!y)}$$

T3
$$\frac{z \subseteq y}{z \subseteq (x!y)}$$

T4
$$\frac{y \subseteq x}{y \subseteq \{x\}k}$$

The rules S1–S7 define what a principal *sees*. A principal *sees* what she receives in a message, what she has initially, what she can extract, and what she can construct. These rules correspond to rules 6–10 in the SVO system:

S1
$$\frac{\mathcal{P} \text{ received } x}{\mathcal{P} \text{ sees } x}$$

S2
$$\frac{\mathcal{P} \text{ initiallyhas } x}{\mathcal{P} \text{ sees } x}$$

S3
$$\frac{\mathcal{P} \text{ sees } x, \quad x \rangle \mathcal{P} \rangle y}{\mathcal{P} \text{ sees } y}$$

S4
$$\frac{\mathcal{P} \text{ sees } x, \quad \mathcal{P} \text{ sees } y}{\mathcal{P} \text{ sees } (x!y)}$$

S5
$$\frac{\mathcal{P} \text{ sees } x, \quad \mathcal{P} \text{ sees } k}{\mathcal{P} \text{ sees } \{x\}k}$$

S6
$$\frac{\mathcal{P} \text{ sees } x, \quad \mathcal{P} \text{ sees } k}{\mathcal{P} \text{ sees } \{x\}^\circ k}$$

$$S7 \quad \frac{\mathcal{P} \text{ sees } +k, \mathcal{P} \text{ sees } -l}{\mathcal{P} \text{ sees } k \text{ DH } l}$$

The relation $\rangle \mathcal{P} \rangle$ indicates what the principal \mathcal{P} can *extract*. This is essentially a special case of the subterm relation, where \mathcal{P} has the necessary keys:

$$E1 \quad \vdash x \rangle \mathcal{P} \rangle x$$

$$E2 \quad \frac{x \rangle \mathcal{P} \rangle z}{(x!y) \rangle \mathcal{P} \rangle z}$$

$$E3 \quad \frac{y \rangle \mathcal{P} \rangle z}{(x!y) \rangle \mathcal{P} \rangle z}$$

$$E4 \quad \frac{x \rangle \mathcal{P} \rangle y, \mathcal{P} \text{ sees } \hat{k}}{\{x\}k \rangle \mathcal{P} \rangle y}$$

Formulating rules such as these in terms of operations (such as extraction) which a principal can perform makes it easier both to construct a set of easily automated rules and to prove the resultant rules sound.

To allow reasoning about **fresh** (recently created) terms, we have the following rules. These will be augmented by situation-specific axioms about the freshness of particular recently created atoms (i.e. nonces, fresh keys, etc.) for particular situations. These rules correspond to rules 16–17 in the SVO system:

$$F1 \quad \frac{\text{fresh } x}{\text{fresh } (x!y)}$$

$$F2 \quad \frac{\text{fresh } y}{\text{fresh } (x!y)}$$

$$F3 \quad \frac{\text{fresh } x}{\text{fresh } \{x\}k}$$

$$F4 \quad \frac{\text{fresh } x}{\text{fresh } \{x\}^\circ k}$$

The next five rules define when a principal can be said to have **said** something (we use the relation **says** when a principal has said something fresh, i.e. recently created). These rules correspond to rules 3, 4, 13, 14 and 18 in the SVO system. We assume that when good keys are used, we can believe that messages come from the principal that they appear to come from. That is to say, any third party that possesses the keys can be trusted not to use them. Also, when symmetric keys are used, a principal can distinguish between what it has sent and what its partner has sent.

This distinguishing between what a principal has sent and what its partner has sent (when symmetric

keys are used) can be accomplished in a number of ways. For example, each principal can keep track of messages that they have sent (if a message is encrypted with a good symmetric key and we didn't send it, then the other party must have). It can also be accomplished by ensuring that messages at different steps of a protocol have different structures (if a message is encrypted with a good symmetric key and isn't of the right form to have come from us, then the other party must have sent it). Finally, it can also (but less desirably) be done by adding a one-bit indicator to a message (for a 0, it comes from the principal who initiated the protocol, and for a 1 it comes from the other party).

We indicate this distinguishing (as in SVO logic) with the notation $\{(y!Q)\}k$ for a message encrypted with k and distinguished as coming from Q :

$$Z1 \quad \frac{\mathcal{P} \text{ said } x, y \subseteq x}{\mathcal{P} \text{ said } y}$$

$$Z2 \quad \frac{\mathcal{P} \text{ says } x, y \subseteq x}{\mathcal{P} \text{ says } y}$$

$$Z3 \quad \frac{\mathcal{P} \text{ said } x, \text{ fresh } x}{\mathcal{P} \text{ says } x}$$

$$Z4 \quad \frac{\mathcal{P} \text{ received } x, x \rangle \mathcal{P} \rangle \{(y!Q)\}k, \mathcal{P} \xleftarrow{k} Q}{Q \text{ said } \{(y!Q)\}k}$$

$$Z5 \quad \frac{\mathcal{P} \text{ received } x, x \rangle \mathcal{P} \rangle \{y\}-l, Q \text{ PK } +l}{Q \text{ said } \{y\}-l}$$

The most important part of our system, as in SVO logic, is reasoning about *belief*. We wish principals to have *trust* in certain properties and conditions: this is in fact the main goal of cryptographic protocols. Trust is in fact a particular kind of belief, and is best analysed in terms of the logic of belief.

We use the following three general rules for the modal logic of belief (technically known as K4 or doxastic logic [13, 14, 15]), corresponding to rules 1 and 2 in the SVO system. Note that not all true statements are necessarily believed, not all beliefs are necessarily true, and we are using the \models symbol to express belief rather than provability (for lack of a better symbol which can be used in both \LaTeX and Isabelle). Note also that \implies means implication:

$$M1 \quad \frac{\mathcal{P} \models a \implies b, \mathcal{P} \models a}{\mathcal{P} \models b}$$

$$M2 \quad \frac{\mathcal{P} \models a}{\mathcal{P} \models \mathcal{P} \models a}$$

M3 $\vdash \mathcal{P} \models a$ where a is a tautology

However, there are a number of areas where a principal believes things if and only if they are true, because of her personal knowledge. This includes objects that the principal has and operations that the principal can perform:

$$\text{B1} \quad \frac{\mathcal{P} \text{ received } x}{\mathcal{P} \models \mathcal{P} \text{ received } x}$$

$$\text{B2} \quad \frac{x \succ \mathcal{P} \succ y}{\mathcal{P} \models x \succ \mathcal{P} \succ y}$$

$$\text{B3} \quad \frac{\mathcal{P} \text{ initiallyhas } x}{\mathcal{P} \models \mathcal{P} \text{ initiallyhas } x}$$

$$\text{B4} \quad \frac{\mathcal{P} \text{ PK } + k}{\mathcal{P} \models \mathcal{P} \text{ PK } + k}$$

$$\text{B5} \quad \frac{\mathcal{P} \text{ sees } x}{\mathcal{P} \models \mathcal{P} \text{ sees } x}$$

We replace rules 11 and 12 of the SVO system by a somewhat more complicated set of rules. This is necessary because the attempt to give $\mathcal{P} \models a$ a meaning unrelated to a can lead to unsoundness. This can occur when there is conflict between one set of rules that define the truth of a and another set of rules that define the truth of $\mathcal{P} \models a$. If the semantics tries to enforce the falseness of $\mathcal{P} \models a$, but $\mathcal{P} \models a$ can be inferred from $\mathcal{P} \models b$ and $b \implies a$, then the rules will be unsound.

We first provide a set of rules that say a principal **understands** something if it can be built up by construction or decryption from atomic elements:

U1 $\mathcal{P} \text{ understands } x \iff \mathcal{P} \text{ sees } x$ for atomic x

$$\text{U2} \quad \frac{\mathcal{P} \text{ understands } x, \mathcal{P} \text{ understands } y}{\mathcal{P} \text{ understands } (x ! y)}$$

$$\text{U3} \quad \frac{\mathcal{P} \text{ understands } x, \mathcal{P} \text{ sees } k}{\mathcal{P} \text{ understands } \{x\}k}$$

$$\text{U4} \quad \frac{\mathcal{P} \text{ understands } x, \mathcal{P} \text{ sees } k}{\mathcal{P} \text{ understands } \{x\}^\circ k}$$

$$\text{U5} \quad \frac{\mathcal{P} \text{ understands } x, \mathcal{P} \text{ sees } \{x\}k, \mathcal{P} \text{ sees } \hat{k}}{\mathcal{P} \text{ understands } \{x\}k}$$

$$\text{B6} \quad \frac{\mathcal{P} \text{ understands } x}{\mathcal{P} \models \mathcal{P} \text{ understands } x}$$

The rules U3 and U5 produce similar conclusions, but for different reasons. With rule U3, \mathcal{P} understands $\{x\}k$ because she can construct it from things she understands, while with rule U5 she understands $\{x\}k$ because she can decrypt it to give something which she understands. In the special case of symmetric keys, where $\hat{k} \equiv k$, U5 becomes a special case of U3.

We also have a relation $\succ \mathcal{P} \succ$ which indicates that a principal recognises something as being a subterm of another (another example of a relation expressed in terms of an operation that a principal can perform). This relation is stronger than the relation \supseteq but weaker than $\succ \mathcal{P} \succ$. The difference lies in the rule E9, which says that a principal can recognise subterms of $\{x\}k$ if she understands $\{x\}k$ through construction (as in rule U3). For example, if h is a one-way hash function (i.e. a key without an inverse) which \mathcal{P} has, and \mathcal{P} understands x , then $\{x\}h \succ \mathcal{P} \succ x$ (we will use this in our first example, later in the paper):

$$\text{E5} \quad \vdash x \succ \mathcal{P} \succ x$$

$$\text{E6} \quad \frac{x \succ \mathcal{P} \succ z}{(x ! y) \succ \mathcal{P} \succ z}$$

$$\text{E7} \quad \frac{y \succ \mathcal{P} \succ z}{(x ! y) \succ \mathcal{P} \succ z}$$

$$\text{E8} \quad \frac{x \succ \mathcal{P} \succ y, \mathcal{P} \text{ sees } \hat{k}}{\{x\}k \succ \mathcal{P} \succ y}$$

$$\text{E9} \quad \frac{x \succ \mathcal{P} \succ y, \mathcal{P} \text{ sees } k, \mathcal{P} \text{ understands } x}{\{x\}k \succ \mathcal{P} \succ y}$$

$$\text{B7} \quad \frac{x \succ \mathcal{P} \succ y}{\mathcal{P} \models x \succ \mathcal{P} \succ y}$$

We use this to provide two variations of rules Z1 and Z2 that use the relation $\succ \mathcal{P} \succ$ to indicate that a principal recognises a subterm relationship. This is necessary because, to ensure soundness, we have given no interpretation to $\mathcal{Q} \models y \subseteq x$:

$$\text{Z6} \quad \frac{\mathcal{Q} \models \mathcal{P} \text{ said } x, x \succ \mathcal{Q} \succ y}{\mathcal{Q} \models \mathcal{P} \text{ said } y}$$

$$\text{Z7} \quad \frac{\mathcal{Q} \models \mathcal{P} \text{ says } x, x \succ \mathcal{Q} \succ y}{\mathcal{Q} \models \mathcal{P} \text{ says } y}$$

Finally, we introduce terms of the form **assert** x which assert a proposition, and say that a principal believes a proposition if she believes it comes from a trustworthy source (these assertions are usually implicit, based on the context of a message in a particular

protocol). This corresponds to the more limited rule 15 in the SVO system, which is only valid if the trusted source does not make an error. We express our version of the rule in terms of belief to explicitly cater for the case where a source is trusted (in the sense that some principals believe it) but can still make false statements:

$$\begin{array}{l}
\text{C1} \quad \frac{Q \models \mathcal{P} \text{ controls } x \\ Q \models \mathcal{P} \text{ says (assert } x)}{Q \models x} \\
\\
\text{C2} \quad \frac{Q \models \mathcal{P} \text{ controlled } x \\ Q \models \mathcal{P} \text{ said (assert } x)}{Q \models x}
\end{array}$$

There are no rules to infer \mathcal{P} **controls** x (i.e. \mathcal{P} can be trusted when she has recently said x) or \mathcal{P} **controlled** x (\mathcal{P} can be trusted when she has ever said x) since these two statements occur only in situation-specific axioms.

Note that if Q trusts a principal who makes false statements, then she will have false beliefs as a result.

3. The Implementation

We have implemented this system in Isabelle firstly by introducing a logic and a set of tactics based on the sequent calculus and on rules derived from the belief rules M1–M3. Secondly we have attempted to produce a terminating proof tactic, by eliminating circularity from the rules. There are two problems here: on the one hand, rules C1 and C2, which are totally unsuitable for automatic backwards proof (because the hypotheses are instances of the conclusion). In the GUI tool we have developed, these rules are applied manually, by clicking a button. They can also be applied automatically (using forwards proof) if the second hypothesis is explicitly listed by the user as an intermediate goal. The other cause of circularity is rules E1–E4 and S1–S7 which depend on each other. We have addressed this by replacing E1–E4 by an equivalent set of rules:

$$\begin{array}{l}
\text{X1} \quad x \rangle \mathcal{P} \rangle y \iff x \rangle \mathcal{P}, [] \rangle y \\
\text{X2} \quad \vdash x \rangle \mathcal{P}, [] \rangle x \\
\text{X3} \quad \frac{\mathcal{P} \text{ sees } k, \quad x \rangle \mathcal{P}, \lambda \rangle x}{x \rangle \mathcal{P}, [k|\lambda] \rangle x} \\
\text{X4} \quad \frac{x \rangle \mathcal{P}, \lambda \rangle z}{(x!y) \rangle \mathcal{P}, \lambda \rangle z}
\end{array}$$

$$\begin{array}{l}
\text{X5} \quad \frac{y \rangle \mathcal{P}, \lambda \rangle z}{(x!y) \rangle \mathcal{P}, \lambda \rangle z} \\
\text{X6} \quad \frac{x \rangle \mathcal{P}, [\hat{k}|\lambda] \rangle y}{\{x\}k \rangle \mathcal{P}, \lambda \rangle y}
\end{array}$$

Here $[k|\lambda]$ represents a list of keys with head k and tail λ , and $[]$ represents an empty list. The notation $x \rangle \mathcal{P}, [k|\lambda] \rangle y$ means that the principal \mathcal{P} can extract y from x , and that λ is a list of keys which are needed to do this. These rules have precisely the same effect as the rules E1–E4, but assist automated reasoning by delaying the search for \mathcal{P} **sees** \hat{k} until we know that we need it. Non-termination can still occur, but *only* in pathological cases where a key occurs in places where it is itself necessary for extraction, for example $\{\hat{k}\}k$ or $(\{\hat{k}_1\}k_2! \{\hat{k}_2\}k_1)$.

More formally, define $k_1 \leq_{\mathcal{P}}^{\circ} k_2$ if and only if in a term e received or initially had by \mathcal{P} , there is a subterm $\{e'\}k_2$ where \hat{k}_1 occurs in e' (i.e. $\hat{k}_1 \subseteq e'$ and $\{e'\}k_2 \subseteq e$), and define $\leq_{\mathcal{P}}$ as the transitive (but not reflexive) closure of $\leq_{\mathcal{P}}^{\circ}$. The relation $k_1 \leq_{\mathcal{P}} k_2$ formalises the concept that k_2 is needed to extract \hat{k}_1 , and hence $k \leq_{\mathcal{P}} k$ formalises the concept that \hat{k} occurs in places where it is itself necessary for extraction.

Now consider the case where, for example, we have \mathcal{P} **received** e , and we are trying to prove \mathcal{P} **sees** \hat{k} by backwards proof using S1 and S3 (which requires $e \rangle \mathcal{P} \rangle \hat{k}$). If it is the case that $k \leq_{\mathcal{P}} k$, backwards proof will not terminate, since application of rules X1–X6 produces \mathcal{P} **sees** \hat{k} as a subgoal. Conversely, if it is *not* the case that $k \leq_{\mathcal{P}} k$, then application of rules X4–X6 results in one of two possible cases:

1. (i) $\{e'\}k$ is not a subterm of e , and rule X6 will not add \hat{k} to the list of keys λ , or
2. (ii) $\{e'\}k$ is a subterm of e for some e' , but \hat{k} (or a key required to find \hat{k}) does not occur in e' and hence rule X3 is never applied for \hat{k} .

In both cases, non-termination will not result. In other words, non-termination occurs if and only if $k \leq_{\mathcal{P}} k$ for some k .

In addition to the rules X1–X6, we also define a *loose* subterm relation, which differs from the *strict* subterm relation (\subseteq) in the last three rules:

$$\begin{array}{l}
\text{T5} \quad \vdash x \sqsubseteq x \\
\text{T6} \quad \frac{z \sqsubseteq x}{z \sqsubseteq (x!y)}
\end{array}$$

$$\text{T7} \quad \frac{z \sqsubseteq y}{z \sqsubseteq (x!y)}$$

$$\text{T8} \quad \frac{y \sqsubseteq x}{y \sqsubseteq \{x\}k}$$

$$\text{T9} \quad \frac{y \sqsubseteq k}{y \sqsubseteq \{x\}k}$$

$$\text{T10} \quad \vdash k \sqsubseteq +k$$

$$\text{T11} \quad \vdash k \sqsubseteq -k$$

If $+k$ and $-k$ are a public/private key pair, k can be considered to be the information from which the pair is derived, or else $k \sqsubseteq +k$ can be interpreted purely syntactically.

The relation \sqsubseteq is used to define a new quantifier $\exists x \sqsubseteq y. a$, equivalent to $\exists x. x \sqsubseteq y \wedge a$. Within the sequent calculus, we can write a terminating and complete tactic for this quantifier (by the obvious induction on the structure of y), which we cannot do for the existential quantifier in general.

Using this new quantifier, we can replace rules Z1–Z7 by eight new derived rules which are more suitable for automatic backwards proof:

$$\text{D1} \quad \frac{\begin{array}{l} \mathcal{P} \text{ received } x \\ \exists y \sqsubseteq x. \exists k \sqsubseteq x. \mathcal{P} \xleftrightarrow{k} Q \\ \wedge x \rangle \mathcal{P} \rangle \{(y!Q)\}k \wedge \mathbf{fresh } y \\ \wedge z \sqsubseteq \{(y!Q)\}k \end{array}}{Q \text{ says } z}$$

$$\text{D2} \quad \frac{\begin{array}{l} \mathcal{P} \text{ received } x \\ \exists y \sqsubseteq x. \exists l \sqsubseteq x. Q \mathbf{PK} +l \wedge x \rangle \mathcal{P} \rangle \{y\}-l \\ \wedge \mathbf{fresh } y \wedge z \sqsubseteq \{y\}-l \end{array}}{Q \text{ says } z}$$

$$\text{D3} \quad \frac{\begin{array}{l} \mathcal{P} \text{ received } x \\ \exists y \sqsubseteq x. \exists k \sqsubseteq x. \mathcal{P} \xleftrightarrow{k} Q \\ \wedge x \rangle \mathcal{P} \rangle \{(y!Q)\}k \\ \wedge z \sqsubseteq \{(y!Q)\}k \end{array}}{Q \text{ said } z}$$

$$\text{D4} \quad \frac{\begin{array}{l} \mathcal{P} \text{ received } x \\ \exists y \sqsubseteq x. \exists l \sqsubseteq x. Q \mathbf{PK} +l \wedge x \rangle \mathcal{P} \rangle \{y\}-l \\ \wedge z \sqsubseteq \{y\}-l \end{array}}{Q \text{ said } z}$$

\mathcal{P} received x

$$\text{D5} \quad \frac{\begin{array}{l} \exists y \sqsubseteq x. \exists k \sqsubseteq x. \mathcal{P} \models \mathcal{P} \xleftrightarrow{k} Q \\ \wedge x \rangle \mathcal{P} \rangle \{(y!Q)\}k \\ \wedge \mathcal{P} \models \mathbf{fresh } y \\ \wedge \{(y!Q)\}k \succ \mathcal{P} \succ z \end{array}}{\mathcal{P} \models Q \text{ says } z}$$

\mathcal{P} received x

$$\text{D6} \quad \frac{\begin{array}{l} \exists y \sqsubseteq x. \exists l \sqsubseteq x. \mathcal{P} \models Q \mathbf{PK} +l \\ \wedge x \rangle \mathcal{P} \rangle \{y\}-l \\ \wedge \mathcal{P} \models \mathbf{fresh } y \\ \wedge \{y\}-l \succ \mathcal{P} \succ z \end{array}}{\mathcal{P} \models Q \text{ says } z}$$

\mathcal{P} received x

$$\text{D7} \quad \frac{\begin{array}{l} \exists y \sqsubseteq x. \exists k \sqsubseteq x. \mathcal{P} \models \mathcal{P} \xleftrightarrow{k} Q \\ \wedge x \rangle \mathcal{P} \rangle \{(y!Q)\}k \\ \wedge \{(y!Q)\}k \succ \mathcal{P} \succ z \end{array}}{\mathcal{P} \models Q \text{ said } z}$$

\mathcal{P} received x

$$\text{D8} \quad \frac{\begin{array}{l} \exists y \sqsubseteq x. \exists l \sqsubseteq x. \mathcal{P} \models Q \mathbf{PK} +l \\ \wedge x \rangle \mathcal{P} \rangle \{y\}-l \\ \wedge \{y\}-l \succ \mathcal{P} \succ z \end{array}}{\mathcal{P} \models Q \text{ said } z}$$

The soundness of these rules was demonstrated by deriving them from Z1–Z7 in Isabelle. Their completeness (with respect to Z1–Z7) can be demonstrated by a simple induction on proof trees built from Z1–Z7. In particular, it is clear that the relations $y \sqsubseteq x$ etc. must hold if e.g. $x \rangle \mathcal{P} \rangle \{(y!Q)\}k$ so that the use of the bounded quantifier does not lose generality.

The sequents we use have the form $a_1, \dots, a_n \vdash b$ where a_1, \dots, a_n are situation-specific axioms, and b is what we want to prove. The sequent calculus proof tactics are simplified by the fact that the statements \mathcal{P} **initiallyhas** x or \mathcal{P} **received** x occur only in situation-specific axioms: they are not created by the rules. Thus they need only be searched for in the left-hand side of a sequent. The sequent calculus version of rule D1 thus has the form (here $\alpha, \beta, \gamma, \delta$ represent sets of statements):

$$\text{D1}' \quad \frac{\alpha, \mathcal{P} \text{ received } x, \beta \vdash \gamma, \exists y \sqsubseteq x. \dots, \delta}{\alpha, \mathcal{P} \text{ received } x, \beta \vdash \gamma, Q \text{ says } z, \delta}$$

The cases of D2–D8 are similar.

4. First Example

In this paper we consider two examples of the use of SVD logic. Our first example involves the encrypted and authenticated transfer of a message M between two parties \mathcal{A} and \mathcal{B} , with the participation of a trusted third party \mathcal{C} . Such a transfer occurs in e.g. electronic commerce. We begin with a series of axioms for this specific situation (we use some fairly obvious shorthand conventions, and h and k are a hash key and a new symmetric session key respectively). We also use $*$ as a wild-card pattern (our GUI tool understands such patterns):

$$\begin{aligned} &\mathcal{A} \text{ PK } +a, \mathcal{B} \text{ PK } +b, \mathcal{C} \text{ PK } +c \\ &\mathcal{A} \text{ initiallyhas } h, k, +a, -a, +c, \mathcal{A}, \mathcal{B}, \mathcal{C} \\ &\mathcal{B} \text{ initiallyhas } h, +b, -b, +c, \mathcal{A}, \mathcal{B}, \mathcal{C} \\ &\mathcal{A} \models \mathcal{C} \text{ PK } +c \\ &\mathcal{B} \models \mathcal{C} \text{ PK } +c \\ &\mathcal{A} \models \mathcal{C} \text{ controlled } * \text{ PK } * \\ &\mathcal{B} \models \mathcal{C} \text{ controlled } * \text{ PK } * \end{aligned}$$

The protocol involves two certificates from \mathcal{C} , which assert the existence and goodness of public keys for \mathcal{A} and \mathcal{B} (authenticated by the private key for \mathcal{C}):

$$\begin{aligned} &\mathcal{A} \text{ received } \theta = \{(+a! (\text{assert } \mathcal{A} \text{ PK } +a))\} -c \\ &\mathcal{A} \text{ received } \{(+b! (\text{assert } \mathcal{B} \text{ PK } +b))\} -c \end{aligned}$$

We use θ as a shorthand for the first certificate, which \mathcal{A} will pass on to \mathcal{B} . Finally there is a complex message from \mathcal{A} to \mathcal{B} , which contains the message M , a digital signature of M , and the certificate θ , all encrypted with the session key k . The session key is attached, encrypted with the public key for \mathcal{B} :

$$\mathcal{B} \text{ received } (\{(M! \{\{M\}h\} -a! \theta)\} k! \{k\} +b)$$

We are then able to prove the following statements (in particular that \mathcal{B} believes that the message M indeed came from \mathcal{A}):

$$\begin{aligned} &\mathcal{A} \models \mathcal{C} \text{ said } (\text{assert } \mathcal{B} \text{ PK } +b) && \text{by D8, E5-E8} \\ &\mathcal{A} \models \mathcal{B} \text{ PK } +b && \text{by C2} \\ &\mathcal{A} \text{ sees } +b && \text{by E1-E4, S1, S3} \\ &\mathcal{B} \text{ sees } k && \text{by E1-E4, S1, S3} \\ &\mathcal{B} \text{ sees } M && \text{by E1-E4, S1, S3} \\ &\mathcal{B} \text{ sees } +a && \text{by E1-E4, S1, S3} \\ &\mathcal{B} \models \mathcal{C} \text{ said } (\text{assert } \mathcal{A} \text{ PK } +a) && \text{by D8, E1-E8} \end{aligned}$$

$$\mathcal{B} \models \mathcal{A} \text{ PK } +a \quad \text{by C2}$$

$$\mathcal{B} \models \mathcal{A} \text{ said } M \quad \text{by D8, E1-E9}$$

The last step uses the E8 and E9 rules, and the fact that \mathcal{B} recognises M inside the digital signature, i.e. $\{\{M\}h\} -a \succ_{\mathcal{B}} M$. For the purposes of this analysis, the message M is considered to be an atom, since it has no constituent parts as far as our logic is concerned.

5. Second Example

Our second example involves the first five steps of the Needham-Schroeder key exchange protocol, as modified to exclude the original design error. Two parties \mathcal{A} and \mathcal{B} wish to communicate privately with each other, using a key generated by a trusted third party \mathcal{C} . The axioms for this specific situation are as follows ($k_{\mathcal{A}\mathcal{C}}$, $k_{\mathcal{B}\mathcal{C}}$, and $k_{\mathcal{A}\mathcal{B}}$ are symmetric keys, and $N_{\mathcal{A}}$ and $N_{\mathcal{B}}$ are nonces, i.e. fresh atomic terms):

$$\begin{aligned} &\mathcal{A} \xleftrightarrow{k_{\mathcal{A}\mathcal{C}}} \mathcal{C}, \mathcal{B} \xleftrightarrow{k_{\mathcal{B}\mathcal{C}}} \mathcal{C}, \mathcal{A} \xleftrightarrow{k_{\mathcal{A}\mathcal{B}}} \mathcal{B} \\ &\mathcal{A} \text{ initiallyhas } k_{\mathcal{A}\mathcal{C}}, N_{\mathcal{A}}, \mathcal{A}, \mathcal{B}, \mathcal{C} \\ &\mathcal{B} \text{ initiallyhas } k_{\mathcal{B}\mathcal{C}}, N_{\mathcal{B}}, \mathcal{A}, \mathcal{B}, \mathcal{C} \\ &\text{fresh } N_{\mathcal{A}}, N_{\mathcal{B}} \\ &\mathcal{A} \models \mathcal{A} \xleftrightarrow{k_{\mathcal{A}\mathcal{C}}} \mathcal{C} \\ &\mathcal{B} \models \mathcal{B} \xleftrightarrow{k_{\mathcal{B}\mathcal{C}}} \mathcal{C} \\ &\mathcal{A} \models \text{fresh } N_{\mathcal{A}} \\ &\mathcal{B} \models \text{fresh } N_{\mathcal{B}} \\ &\mathcal{A} \models \mathcal{C} \text{ controls } * \xleftrightarrow{*} * \\ &\mathcal{B} \models \mathcal{C} \text{ controls } * \xleftrightarrow{*} * \end{aligned}$$

The principal \mathcal{A} initiates the protocol with a message to \mathcal{B} . She responds with a message containing a nonce $N_{\mathcal{B}}$, which is then passed on to the trusted third party, along with a nonce for \mathcal{A} :

$$\begin{aligned} &\mathcal{B} \text{ received } \mathcal{A} \\ &\mathcal{A} \text{ received } \{(\mathcal{A}! N_{\mathcal{B}})\} k_{\mathcal{B}\mathcal{C}} \\ &\mathcal{C} \text{ received } (\mathcal{A}! \mathcal{B}! N_{\mathcal{A}}! \{(\mathcal{A}! N_{\mathcal{B}})\} k_{\mathcal{B}\mathcal{C}}) \end{aligned}$$

This is followed by a complex message from the trusted third party to \mathcal{A} , part of which is in turn passed on to \mathcal{B} . This message includes the key $k_{\mathcal{A}\mathcal{B}}$, an assertion as to its goodness, and the appropriate nonces to guarantee freshness:

$$\theta = \{(N_{\mathcal{B}}! k_{\mathcal{A}\mathcal{B}}! \text{assert } \mathcal{A} \xleftrightarrow{k_{\mathcal{A}\mathcal{B}}} \mathcal{B}! \mathcal{C})\} k_{\mathcal{B}\mathcal{C}}$$

\mathcal{A} received $\{(N_{\mathcal{A}}!k_{\mathcal{A}\mathcal{B}}!\text{assert } \mathcal{A} \xleftrightarrow{k_{\mathcal{A}\mathcal{B}}} \mathcal{B}!\theta!C)\}k_{\mathcal{A}\mathcal{C}}$
 \mathcal{B} received θ

We are then able to prove that both parties have the key $k_{\mathcal{A}\mathcal{B}}$ (and believe that it is good):

\mathcal{A} sees $k_{\mathcal{A}\mathcal{B}}$ by E1–E4, S1, S3
 $\mathcal{A} \models \mathcal{C}$ says (assert $\mathcal{A} \xleftrightarrow{k_{\mathcal{A}\mathcal{B}}} \mathcal{B}$) by D5, E1–8, F1–4
 $\mathcal{A} \models \mathcal{A} \xleftrightarrow{k_{\mathcal{A}\mathcal{B}}} \mathcal{B}$ by C1
 \mathcal{B} sees $k_{\mathcal{A}\mathcal{B}}$ by E1–E4, S1, S3
 $\mathcal{B} \models \mathcal{C}$ says (assert $\mathcal{A} \xleftrightarrow{k_{\mathcal{A}\mathcal{B}}} \mathcal{B}$) by D5, E1–8, F1–4
 $\mathcal{B} \models \mathcal{A} \xleftrightarrow{k_{\mathcal{A}\mathcal{B}}} \mathcal{B}$ by C1

6. The GUI Tool

We have developed a GUI tool for this logic which we call C3PO. A screen dump of the tool is included in the appendix. The tool has been developed in the **Tcl** language using the **TK** graphical toolkit. The tool maintains a list of propositions a_1, \dots, a_n which include the situation-specific axioms and the statements that have been proved so far. When the user enters a statement b to be proved, the sequent $a_1, \dots, a_n \vdash b$ is passed to Isabelle for proving. The **Expect** interfacing extensions to **Tcl/TK** were used to handle communication with Isabelle.

The tool also allows automatic processing of a to-do list of statements, producing a log of successes and failures. Pattern-matching within the tool is used to implement the C1 and C2 rules, and these are applied automatically when a suitable sub-goal is proved. For example, the lists of statements in the two examples above can be used as to-do lists, and can be successfully processed by the tool without any manual intervention by the user. That is, once the to-do list is opened, the only thing the user needs to do is click the **Start Proof** button. If desired, the user can also enter additional statements and attempt to prove them. The time taken to process most examples is a few minutes.

7. Semantics

We prove the soundness of our system using Kripke (possible worlds) semantics [12, 13, 14, 15]. Each (possible) world contains 12 components. Some components are indexed by principals (\mathcal{P}) or time ($t \geq 0$ an integer, since we consider time to increase in fixed steps every time a significant event occurs). The rules

K1–K7 and T1–T11 continue to apply to terms. The components are:

1. Sets N_t of atoms (nonces) created at time t .
2. Sets F_t of terms which are fresh at time t . We assume times up to a fixed limit Δ are fresh. These sets are defined by (i.e. are minimal sets satisfying):

$$\begin{aligned} N_t, N_{t-1}, \dots, N_{t-\Delta} &\subseteq F_t \\ x \in F_t &\implies (x!y) \in F_t, (y!x) \in F_t \\ x \in F_t &\implies \{x\}k \in F_t, \{x\}^\circ k \in F_t \end{aligned}$$

The statement **fresh** x is true in world w at time t (we use the notation $\langle w, t \rangle : \text{fresh } x$) if and only if $x \in F_t$, where the world w is a 12-tuple of the form $(\{N_t\}, \{F_t\}, M, \{S_{\mathcal{P}t}\}, \{\hookrightarrow_{\mathcal{P}t}\}, \{G_t\}, \{H_t\}, \{Z_{\mathcal{P}t}\}, \{U_{\mathcal{P}t}\}, \{\mapsto_{\mathcal{P}t}\}, \{C_{\mathcal{P}t}\}, \{D_{\mathcal{P}t}\})$. It is clear that the freshness rules F1–F4 are sound on this interpretation.

3. A message history M containing elements of the form $(\mathcal{P} \xrightarrow{t} \mathcal{Q}: x), \mathcal{P} \neq \mathcal{Q}$ (\mathcal{P} sends x to \mathcal{Q} at time t). We define $\langle w, t \rangle : \mathcal{Q}$ received x if and only if $(\mathcal{P} \xrightarrow{t'} \mathcal{Q}: x) \in M$ for some $t' < t$ (next-step receipt).
4. Sets $S_{\mathcal{P}t}$ of terms seen by \mathcal{P} at time t . These sets are defined by the choice of $S_{\mathcal{P}0}$ and:

$$\begin{aligned} S_{\mathcal{P}t'} &\subseteq S_{\mathcal{P}t} \quad \text{for } t' < t \\ x \in S_{\mathcal{P}t}, y \in S_{\mathcal{P}t} &\implies (x!y) \in S_{\mathcal{P}t} \\ x \in S_{\mathcal{P}t}, k \in S_{\mathcal{P}t} &\implies \{x\}k \in S_{\mathcal{P}t}, \{x\}^\circ k \in S_{\mathcal{P}t} \\ +k \in S_{\mathcal{P}t}, -l \in S_{\mathcal{P}t} &\implies k \text{ DH } l \in S_{\mathcal{P}t} \\ x \in S_{\mathcal{P}t}, x \hookrightarrow_{\mathcal{P}t} y &\implies y \in S_{\mathcal{P}t} \\ (\mathcal{P} \xrightarrow{t'} \mathcal{Q}: x) \in M, t' < t &\implies x \in S_{\mathcal{P}t'}, x \in S_{\mathcal{Q}t} \end{aligned}$$

5. Reflexive and transitive binary relations $\hookrightarrow_{\mathcal{P}t}$ describing what can be extracted at time t , defined by:

$$\begin{aligned} (x!y) &\hookrightarrow_{\mathcal{P}t} x \\ (x!y) &\hookrightarrow_{\mathcal{P}t} y \\ \{x\}k &\hookrightarrow_{\mathcal{P}t} x \quad \text{if } \hat{k} \in S_{\mathcal{P}t} \end{aligned}$$

We define $\langle w, t \rangle : \mathcal{P}$ initiallyhas x if and only if $x \in S_{\mathcal{P}0}$ i.e. in the initial world. We also define $\langle w, t \rangle : \mathcal{P}$ sees x if and only if $x \in S_{\mathcal{P}t}$, and we define $\langle w, t \rangle : x \rangle \mathcal{P} \rangle y$ if and only if $x \hookrightarrow_{\mathcal{P}t} y$. It is clear that the rules E1–E4 and S1–S7 are sound on this interpretation.

6. Sets G_t of triples $(\mathcal{P}, k, \mathcal{Q})$ of symmetric keys which are good at time t . The sets must satisfy our assumptions about how good keys are used:

- (a) $(\mathcal{P}, k, \mathcal{Q}) \in G_t$ if and only if $(\mathcal{Q}, k, \mathcal{P}) \in G_t$.
- (b) If $(\mathcal{P}, k, \mathcal{Q}) \in G_t$ then every term of the form $\{(x! \mathcal{P})\}k$ first occurs in the message history M as $(\mathcal{P} \rightarrow_{t'} \mathcal{R}: e)$ for some \mathcal{R} and $e \supseteq \{(x! \mathcal{P})\}k$.

We define $\langle w, t \rangle : \mathcal{P} \xleftrightarrow{k} \mathcal{Q}$ if and only if $(\mathcal{P}, k, \mathcal{Q}) \in G_t$. It is clear that the rule K8 is sound on this interpretation.

7. Sets H_t of pairs (\mathcal{P}, k) of public keys which are good at time t . These sets must satisfy our assumptions about how good keys are used:

- (a) If $(\mathcal{P}, k) \in H_t$ then every term of the form $\{x\}-k$ first occurs in the message history M as $(\mathcal{P} \rightarrow_{t'} \mathcal{R}: e)$ for some \mathcal{R} and $e \supseteq \{x\}-k$.
- (b) If $(\mathcal{P}, k) \in H_t$ and $(\mathcal{Q}, l) \in H_t$ then $(\mathcal{P}, k \text{ DH } l, \mathcal{Q}) \in G_t$

We define $\langle w, t \rangle : \mathcal{P} \text{ PK } +k$ if and only if $(\mathcal{P}, k) \in H_t$. It is clear that the rule K9 is sound on this interpretation.

8. Sets $Z_{\mathcal{P}t}$ of terms said by \mathcal{P} at time t , such that $x \in Z_{\mathcal{P}t}$ if and only if there is some term $e \supseteq x$ such that e first occurs in the message history M as $(\mathcal{P} \rightarrow_{t'} \mathcal{Q}: e')$ for some \mathcal{Q} , $e' \supseteq e$, and $t' < t$. We define $\langle w, t \rangle : \mathcal{P} \text{ said } x$ if and only if $x \in Z_{\mathcal{P}t}$, and $\langle w, t \rangle : \mathcal{P} \text{ says } x$ if only if $x \in Z_{\mathcal{P}t} \cap F_t$. Soundness of the rules Z1 and Z2 follow from the $e \supseteq x$ condition. Soundness of the rule Z3 follows from the reference to F_t . Soundness of the rules Z4 and Z5 follow from the definitions of G_t and H_t .

9. Sets $U_{\mathcal{P}t}$ of terms understood by \mathcal{P} at time t . These sets are defined by:

$$U_{\mathcal{P}t'} \subseteq U_{\mathcal{P}t} \quad \text{for } t' < t$$

$$x \in U_{\mathcal{P}t} \iff x \in S_{\mathcal{P}t} \quad \text{for atomic } x$$

$$x \in U_{\mathcal{P}t}, y \in U_{\mathcal{P}t} \implies (x!y) \in U_{\mathcal{P}t}$$

$$x \in U_{\mathcal{P}t}, k \in S_{\mathcal{P}t} \implies \{x\}k \in U_{\mathcal{P}t}, \{x\}^\circ k \in U_{\mathcal{P}t}$$

$$x \in U_{\mathcal{P}t}, \{x\}k \in S_{\mathcal{P}t}, \hat{k} \in S_{\mathcal{P}t} \implies \{x\}k \in U_{\mathcal{P}t}$$

We define $\langle w, t \rangle : \mathcal{P} \text{ understands } x$ if and only if $x \in U_{\mathcal{P}t}$. It is clear that the rules U1–U5 are sound on this interpretation. It also follows from the definition of $S_{\mathcal{P}t}$ that $U_{\mathcal{P}t} \subseteq S_{\mathcal{P}t}$.

10. Reflexive and transitive binary relations $\mapsto_{\mathcal{P}t}$ describing what can be recognised as a subterm at time t , defined by:

$$(x!y) \mapsto_{\mathcal{P}t} x$$

$$(x!y) \mapsto_{\mathcal{P}t} y$$

$$\{x\}k \mapsto_{\mathcal{P}t} x \quad \text{if } \hat{k} \in S_{\mathcal{P}t}$$

$$\{x\}k \mapsto_{\mathcal{P}t} x \quad \text{if } k \in S_{\mathcal{P}t} \text{ and } x \in U_{\mathcal{P}t}$$

We define $\langle w, t \rangle : x \succ_{\mathcal{P}} y$ if and only if $x \mapsto_{\mathcal{P}t} y$. It is clear that the rules E5–E9 are sound on this interpretation.

11. Sets $C_{\mathcal{P}t}$ of statements controlled when fresh by \mathcal{P} at time t . We define $\langle w, t \rangle : \mathcal{P} \text{ controls } x$ if and only if $x \in C_{\mathcal{P}t}$. The relation of these sets to the rule C1 will be considered when we discuss the accessibility relation below.

12. Sets $D_{\mathcal{P}t}$ of statements controlled even when not fresh by \mathcal{P} at time t . We define $\langle w, t \rangle : \mathcal{P} \text{ controlled } x$ if and only if $x \in D_{\mathcal{P}t}$.

The semantics of belief is handled in the normal way by defining a transitive (but not reflexive) *accessibility* relation $\sim_{\mathcal{P}}$. We write $w \sim_{\mathcal{P}} w'$ to mean that the possible world w' is consistent with the beliefs of principal \mathcal{P} in world w (i.e. if \mathcal{P} is really in world w , with certain beliefs, then w' is one of the worlds that she thinks she might be in). We require that this relation leaves unchanged $S_{\mathcal{P}t}$, $\hookrightarrow_{\mathcal{P}t}$, $U_{\mathcal{P}t}$ and $\mapsto_{\mathcal{P}t}$ for every t , as well as all pairs (\mathcal{P}, k) in H_t for every t , and all messages to or from \mathcal{P} in M (with the proviso that messages to \mathcal{P} in M may have a different sender). We also require that for statements a about the world, if $a \in C_{\mathcal{P}t}$ in w' and $(\text{assert } a) \in Z_{\mathcal{P}t} \cap F_t$ in w' then a is true about the world w' at time t , i.e. $\langle w', t \rangle : a$. Similarly if $a \in D_{\mathcal{P}t}$ in w' and $(\text{assert } a) \in Z_{\mathcal{P}t}$ in w' then $\langle w', t \rangle : a$.

We define (in the normal way) $\langle w, t \rangle : \mathcal{P} \models a$ if and only if for every world w' such that $w \sim_{\mathcal{P}} w'$, $\langle w', t \rangle : a$. It is clear that the rules B1–B7 and C1–C2 are sound on this interpretation, and the rules M1–M3 are known to be sound for a semantics of this kind. Soundness of the rules Z6 and Z7 follows from the fact that $x \mapsto_{\mathcal{P}t}$ implies $x \supseteq y$. Note that we can have $\langle w, t \rangle : \mathcal{P} \models \text{false}$ if there is no world w' such that $w \sim_{\mathcal{P}} w'$.

With the semantics we have chosen, the rules are not complete. In particular, we may not be able to prove $\mathcal{P} \models a$ even if a is true in every accessible world. For example, we may have $x \supseteq y$, but we cannot prove $\mathcal{P} \models x \supseteq y$, so we would use rules Z6 and Z7 which require proving $x \succ_{\mathcal{P}} y$. We can think of $x \succ_{\mathcal{P}} y$ as meaning that there is *evidence* to convince \mathcal{P} that

$x \supseteq y$. We could have made the semantics match the rules more closely (by including as a 13th component of the world a relation which is sometimes \supseteq and sometimes $\mapsto p_t$). However, since our goal was to prove soundness, the semantics we have chosen is sufficient. We also feel that the question of whether there is sufficient evidence for a principal to believe something is more easily tackled proof-theoretically, by means of rules such as E1–E9, rather than model-theoretically.

8. Related Work

Prior work in automating protocol objects has followed two approaches differing from ours. Automation in Prolog (e.g. [5]) can provide a useful tool, but since correctness of the results depends on correctness of the complex Prolog program, the results cannot necessarily be treated with total confidence. Automation using a theorem prover like HOL (e.g. [6, 7, 8, 9, 10]) or Isabelle is more promising and has led to useful and powerful tools [7] (although no soundness proof appears to exist for this rule set). However, the use of complex proof tactics can sometimes fail to prove true theorems (especially within belief logic, as discussed in [6]).

In contrast, our approach has been to simplify automation by careful formulation of the rules, in particular the rules X1–X6 and the derived rules D1–D8. With these rules a relatively simple and generic proof tactic can be used while ensuring termination (although, as noted in section 3 above, non-termination is still possible for certain pathological cases). We believe our approach is also more easily generalised to new protocol analysis objects that may be developed in the future.

9. Conclusion

We have presented an improved version of BAN logic which is easily automated and proved sound. We have developed an implementation using Isabelle and a GUI interface. Although we have many more rules than SVO logic (66 rather than 20), the rules are relatively simple (the more complex rules D1–D8 are derived) and hence easier to automate and prove correct. Some of the complexity has been forced by the need to avoid looping in automatic proof tactics, and some by the need to ensure that $\mathcal{P} \models a$ does not have a meaning unrelated to a , since this can lead to unsoundness. We have applied our tool to a number of examples relevant to electronic commerce. The fact that our rules have been proved sound provides us with confidence that the results of our tool are meaningful. This is particularly important because the development of our

tool was intended to assist in conducting information security evaluation of cryptographic protocols at the ITSEC E6 level.

However, our logic is not suitable for proving *negative* results (e.g. $\sim \mathcal{E} \text{ sees } k$, where \mathcal{E} is an eavesdropper). Results of this kind, which are practically of great value, are probably best achieved with a model-theoretic tool, since the absence of a proof does not generally mean that a statement is false.

10. Acknowledgements

The author is deeply indebted to Bernard Colbert, Brendan Mahony, and several anonymous referees for comments on earlier versions of this paper.

References

- [1] M. Burrows, M. Abadi and R. Needham, *A Logic of Authentication*, Research Report 39, Digital Systems Research Centre, February 1989
- [2] M. Abadi and M. R. Tuttle, *A Semantics for a Logic of Authentication*, Proceedings of the Tenth ACM Symposium on Principles of Distributed Computing, August 1991, 201–216
- [3] L. Gong, R. Needham and R. Yahalom, *Reasoning about Belief in Cryptographic Protocols*, Proceedings of the 1990 IEEE Computer Society Symposium on Security and Privacy, 234–248
- [4] A. Mathuria, R. Safavi-Naini and P. Nickolas, *Some Remarks on the Logic of Gong, Needham and Yahalom*, Proceedings of the 1994 International Computer Symposium, December 1994, 305–308.
- [5] A. Mathuria, R. Safavi-Naini and P. Nickolas, *On the Automation of GNY logic*, Proceedings of the 1995 Australian Computer Conference, January–February 1995, 370–379.
- [6] S. Brackin, *An Interface Specification Language for Automatically Analyzing Cryptographic Protocols*, Internet Society Symposium on Network and Distributed System Security, February 1997
- [7] R. Lichota, G. Hammonds and S. Brackin, *Verifying Cryptographic Protocols for Electronic Commerce*, Proceedings of the Second USENIX Workshop on Electronic Commerce, November 1996

- [8] S. Brackin, *Automated Formal Analyses of Cryptographic Protocols*, Proceedings of the 19th National Conference on Information Systems Security, October 1996
- [9] S. Brackin, *A HOL Extension of GNY for Automatically Analyzing Cryptographic Protocols*, Proceedings of Computer Security Foundations Workshop IX, June 1996
- [10] S. Brackin, *Deciding Cryptographic Protocol Adequacy with HOL*, Higher Order Logic Theorem Proving and Its Applications, September 1995
- [11] P. F. Syverson and P. C. van Oorschot, *On Unifying Some Cryptographic Protocol Logics*, Proceedings of the 1994 IEEE Computer Society Symposium on Security and Privacy, May 1994
- [12] K. A. Bowen, *Model Theory for Modal Logic*, D. Reidel, Dordrecht, 1979
- [13] M. Fitting, *Proof Methods for Modal and Intuitionistic Logics*, D. Reidel, Dordrecht, 1983
- [14] J.-J. Ch. Meyer et al, *Epistemic Logic for Computer Science: A Tutorial (Parts 1 and 2)*, EATCS Bulletin **44** (June 1991, 242–270) and **45** (Oct 1991, 256–287)
- [15] L. A. Wallen, *Automated Deduction in Nonclassical Logics*, MIT Press, 1990