Towards the Attacker's View of Protocol Narrations (or, How to Compile Security Protocols)

Zhiwei Li
Department of SIS
UNC Charlotte
Charlotte, NC 28223
zli19@uncc.edu

Weichao Wang
Department of SIS and CyberDNA
UNC Charlotte
Charlotte, NC 28223
weichaowang@uncc.edu

ABSTRACT

As protocol narrations are widely used to describe security protocols, efforts have been made to formalize or devise semantics for them. An important, but largely neglected, question is whether or not the formalism faithfully accounts for the attacker's view. Several attempts have been made in the literature to recover the attacker's view. They, however, are rather restricted in scope and quite complex. This greatly impedes the ability of protocol verification tools to detect intricate attacks.

In this paper, we establish a faithful view of the attacker based on rigorous, yet intuitive, interpretations of exchanged messages. This gives us a new way to look at attacks and protocol implementations. Specifically, we identify two types of attacks that can be thawed through adjusting the protocol implementation; and show that such an ideal implementation does not always exist. Overall, the obtained attacker's view provides a path to more secure protocol designs and implementations.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Protocol verification*; D.2.4 [**Software**]: Software/Program Verification—*Formal methods*

General Terms

Security

1. INTRODUCTION

Although protocol narrations are widely used in security literature to describe security protocols, different groups of people view the informal description rather differently. Such a discrepancy among them makes it extremely difficult to evaluate security properties of a protocol.

First, the designer's view of protocol narrations is often "optimistic", because the expected protocol execution naturally leads designers to ignore other possible protocol executions. As an example, let us consider the following Otway-Rees protocol [32].

1. $A \to B : M, A, B, \{N_a, M, A, B\}_{K_{as}}$
2. $B \to S : M, A, B, \{N_a, M, A, B\}_{K_{as}}, \{N_b, M, A, B\}_{K_{bs}}$
3. $S \to B : M, \{N_a, K_{ab}\}_{K_{as}}, \{N_b, K_{ab}\}_{K_{bs}}$
4. $B \to A : M, \{N_a, \underline{K_{ab}}\}_{K_{as}}$

Here, A, B, and S denote different roles of the protocol, and the sequence of message exchanges illustrates the intended execution trace of the protocol. It is expected that at the last step A would receive a symmetric key K_{ab}, whereas A could be cheated to accept (M, A, B) as the symmetric key in a well-known type flaw attack [14].

Second, the implementor's view of protocol narrations can be "pessimistic", because how principals check incoming messages is often neglected in protocol narrations [1]. That is to say, implementors may unnecessarily treat some incoming messages as "black-boxes" and thus allow protocol executions that are not in compliance with the protocol narrations [13]. For example, Ceelen et al. [12] show that Lowe's modified KSL protocol [27] is subject to the selected-name attack. This attack arises because the implementation fails to check an agent's name, which could have been implied by the protocol narration.

There is little point in pretending that a protocol will only execute in accordance with the designer's view. If we adopt the optimistic view in our analysis, attacks that are not in accordance with this view will never be found, such as the type flaw attack on the Otway-Rees protocol. On the contrary, if we adopt the pessimistic view, spurious attacks may be detected due to the absence of some necessary condition checks.

In this paper, we address this discrepancy by establishing a faithful attacker's view of protocol narrations. The view is "faithful" in a sense that all, and only, protocol executions in compliance with a given protocol narration are identified, as shown in Figure 1. Unlike most previous work which has focused on formalization or compilation [11, 10, 9, 30], we aim at a semantics that accounts for the most minute aspects of the protocol in the same manner of an attacker. Such a view coincides with a realistic designer's view and a proactive implementor's view.

Overview.

The main challenge of recovering the attacker's view is to determine exactly to what extent an incoming message can be interpreted by a protocol participant. This task relates closely to specifying a participant's internal action(s) (i.e., condition check), which is an essential but largely neglected

ASIACCS '12, May 2–4, 2012, Seoul, Korea.

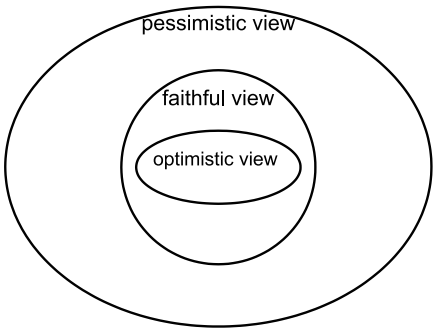


Figure 1: Sets of possible protocol execution traces under different views of a protocol narration

part of protocol specification [1]. Although efforts have been devoted to make such checks explicit, it is far from clear that *all* necessary checks are found. Besides, most of the approaches are specialized for the Dolev-Yao style primitives, and rely on exhaustive case-by-case analysis, without intuitive justifications. To identify *all* necessary internal actions, we provide an intuitive, yet rigorous, justification for checks performed by a principal. Specifically, we extend the notion of *recognizability* [23] to ascertain the extent to which message(s) could be understood. Consequently, we reduce the problem of extraction of semantics from a protocol narration to that of deciding recognizability, of which the decision procedure under Dolev-Yao model is implemented in [24].

We then use this ideal semantics to guide protocol implementation by deriving *all* necessary equality checks. Similar to [13], such implementations are said to be *prudent*. Remarkably, an attack scenario may be useful to refine a protocol implementation; we include additional inequality checks in a refined implementation to prevent the attack. For example, the type flaw attack on the Otway-Reese protocol is infeasible if A checks whether or not the last incoming message is the same as $M, \{N_a, \underline{M, A, B}\}_{K_{as}}$.

Contributions.

The main contributions of this paper are the following:

- We establish a faithful view of the attacker by rigorously examining each participant's ability or inability to cope with potentially ambiguous incoming messages.
- Independent of the attacker model, we present a procedure to extract from a given protocol narration its ideal semantics. This procedure boils down to deciding recognizability, for which decidability results are known under the standard Dolev-Yao model [24].
- We propose a novel classification of protocol implementations and attacks according to the attacker's view. Specifically, we prove that an ideal implementation does not always exist, and thereby design a procedure to derive a prudent implementation to approach it, which performs all necessary equality checks.
- In light of the new classification, we propose a semi-automated implementation refinement paradigm that highlights inequality checks to thwart type-II attacks (defined in Section 5.3). As the new implementation cannot be achieved either by the protocol designers or by the protocol verifiers alone, we motivate the interplay between protocol design and verification via the semi-automated refinement process.

Organization.

The remainder of this paper is organized as follows: Section 2 introduces background materials. Section 3 is dedicated to the interpretations of exchanged messages in protocol narrations. Section 4 gives the ideal semantics of protocol narrations based on interpretations of the exchanged messages. In light of this semantics, Section 5 presents our classification of protocol implementations and attacks. Section 6 discusses related work. Section 7 concludes the paper and outlines the future work.

2. PRELIMINARIES

In this section, we briefly review the basic definitions of term algebra, equational theory, and deducibility.

A *signature* is a finite set of function symbols $\mathcal{F}$ and a possibly infinite set of constants $\mathcal{A}$. Each function symbol has an associated arity. We discriminate *public* and *private* function symbols, respectively denoted by $\mathcal{F}^+$ and $\mathcal{F}^-$. We define the *term algebra* $\mathcal{T}(\mathcal{F}, \mathcal{A}, \mathcal{X})$ as the smallest set containing $\mathcal{X}$ and $\mathcal{A}$ such that $f(t_1, \cdots, t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{A}, \mathcal{X})$ whenever $f \in \mathcal{F}$ with arity n, and $t_1, \cdots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{A}, \mathcal{X})$. Elements of the set $\mathcal{T}(\mathcal{F}, \mathcal{A}, \mathcal{X})$ are called *terms*. To avoid confusion, syntactic equality of two terms t_1 and t_2 will be denoted by $t_1 =_s t_2$. As usual, $fv(t)$ is defined as the set of variables that occur in term t. A term is *ground* if $fv(t) = \emptyset$. We tend to use the words "term" and "message" interchangeably in the rest of this paper.

An *equation* is a pair of terms, written $s = t$ and an *equational theory* E is presented by a finite set of equations. We write $t_1 =_E t_2$ when equation $t_1 = t_2$ is a logical consequence of E. A *substitution* is a finite tuple $[t_1/x_1, ..., t_n/x_n]$ mapping from variables x_i to terms t_i. The *domain* and *range* of a substitution σ are defined by $Dom(\sigma) \stackrel{def}{=} \{x | x\sigma \neq_s x\}$ and $Ran(\sigma) \stackrel{def}{=} \bigcup_{x \in Dom(\sigma)} \{x\sigma\}$, respectively. We write $\sigma = \theta$ if $Dom(\sigma) = Dom(\theta)$ and $x\sigma =_s x\theta$ for all x. We define the *composition* of substitutions σ and θ as a new substitution $\sigma \circ \theta$ such that $t\sigma \circ \theta =_s (t\sigma)\theta$. We use ϵ to denote an empty substitution, that is $Dom(\epsilon) = \emptyset$.

Let E be an equational theory and X a set of variables. We say that substitution σ is more general modulo E on X than the substitution θ, and write $\sigma \preceq_E^X \theta$, if there exists a substitution λ such that $x\theta =_E x\sigma\lambda$ for all $x \in X$.

The most straightforward way to model a principal's knowledge is in terms of message deducibility [18, 26]. That is, given an equational system E and some messages T one might be able to compute another message t from T under equational theory E. Formally,

$\boxed{\vdash}$

(R1) $\dfrac{t \in T}{T \vdash t}$

(R2) $\dfrac{T \vdash t_1 \cdots T \vdash t_k}{T \vdash f(t_1, \cdots, t_k)}\ f \in \mathcal{F}^+$

$\boxed{\vdash_E}$

(R3) $\dfrac{T \vdash s \quad s =_E t}{T \vdash_E t}$

We use the following equational theory E_{dy} to model the standard Dolev-Yao intruder [18].

$\mathcal{F}^+_{dy}$	$\mathtt{pair}, \mathtt{senc}, \mathtt{penc}, \mathtt{hash}$
	$\mathtt{fst}, \mathtt{snd}, \mathtt{sdec}, \mathtt{pdec}$
$\mathcal{F}^-_{dy}$	$\mathtt{pk}, \mathtt{sk}$
E_{dy}	$\mathtt{fst}(\mathtt{pair}(x,y)) = x$
	$\mathtt{snd}(\mathtt{pair}(x,y)) = y$
	$\mathtt{sdec}(\mathtt{senc}(x,y),y) = x$
	$\mathtt{pdec}(\mathtt{penc}(x,\mathtt{pk}(y)),\mathtt{sk}(y)) = x$
	$\mathtt{pdec}(\mathtt{penc}(x,\mathtt{sk}(y)),\mathtt{pk}(y)) = x$

To reduce notational clutter, we use K_a^+, K_a^-, and $s \cdot t$ as shorthands for $\mathtt{pk}(A)$, $\mathtt{sk}(A)$, and $\mathtt{pair}(s,t)$, respectively. Besides, we use $t_1 \cdot t_2 \cdot t_3 \cdot \cdots \cdot t_n$ to denote $(((t_1 \cdot t_2) \cdot t_3) \cdot \cdots \cdot t_n)$. Additionally, $\{s\}_t$ denotes $\mathtt{penc}(s,t)$ if t is either a public/private key, and $\mathtt{senc}(s,t)$ otherwise.

Proposition 2.1. *Let T be a term set and σ be a substitution. Then, $T\sigma \vdash_E t$ if and only if $T \vdash t'$ for some t' such that $t'\sigma =_E t$. Such a term t' is called a* recipe *of t.*

3. INTERPRETING INCOMING MESSAGES

In this section we show how to interpret exchanged messages in protocol narrations. The presentation proceeds in three steps. First, we introduce a new knowledge representation *markup term set* to account for uncertainty. Then, we present an operational equivalence relation to capture one's *inability* to distinguish two interpretations of a message. Finally, we use recognizability to precisely characterize one's *ability* to interpret an incoming message.

3.1 Accounting For Uncertainty

In a hostile protocol execution environment, an incoming message almost always has some part(s) being ambiguous. For example, in the Otway-Rees protocol after exchanging the first three messages, principal A is expecting K_{AB} from the trusted third party S. However, since K_{AB} is dynamically generated, A is uncertain about its value, and thus will accept any bit string of the same length.

To account for uncertainty, we introduce the following notion to encapsulate one's epistemic state.

Definition 3.1 (Markup Term Set)**.** *A* markup term set*, notated as $\vec{T}$, is a pair $\langle T, \sigma\rangle$, where σ is a substitution such that $Dom(\sigma) \subseteq fv(T)$.*

Here, a variable stands for an ambiguous (part of) message, and a ground substitution corresponds to one possible interpretation. This definition accords with the possible worlds semantics for knowledge [21], in which the principal's actual epistemic state resides in one of many possible states, and all possible states are indistinguishable to the principal. We will write $\vec{T}\downarrow_{ts}$ and $\vec{T}\downarrow_{subs}$ for the term set and substitution in $\vec{T}$, respectively.

Example 1. *Let us use markup term sets to model a principals' knowledge after completion of the Otway-Rees protocol. The initial knowledge of A, B, and C is given by T_{a0}, T_{b0}, and T_{s0}, respectively, where*

$$T_{a0} = \{M, A, B, S, N_a, K_{as}\}$$
$$T_{b0} = \{A, B, S, N_b, K_{bs}\}$$
$$T_{s0} = \{A, B, S, K_{as}, K_{bs}\}$$

Upon completion of the protocol, the knowledge of each principal becomes

$$\vec{T_a} = \langle T_{a0} \cup \{x_4\}, [\{N_a \cdot K_{ab}\}_{K_{as}}/x_4]\rangle$$
$$\vec{T_b} = \langle T_{b0} \cup \{x_1, x_3\}, [(M \cdot A \cdot B \cdot \{N_a \cdot M \cdot A \cdot B\}_{K_{as}})/x_1, \{N_b \cdot K_{ab}\}_{K_{bs}})/x_3]\rangle$$
$$\vec{T_s} = \langle T_{s0} \cup \{x_2\}, [(A \cdot B \cdot \{N_a \cdot M \cdot A \cdot B\}_{K_{as}}, \{N_b \cdot M \cdot A \cdot B\}_{K_{bs}})/x_2]\rangle$$

where x_1 to x_4 indicate the four incoming messages.

3.2 Operational Equivalence

We have used a markup term set to represent a principal's epistemic state. Now, we introduce an indistinguishability relation to capture one's inability to discriminate two epistemic states.

Definition 3.2 (Operational Equivalence)**.** *Let T be a term set and σ_1 and σ_2 be two substitutions such that $Dom(\sigma_1)$, $Dom(\sigma_2) \subseteq fv(T)$. They are* operationally equivalent *in equational theory E w.r.t. term set T, written as $\sigma_1 \approx_{E,T} \sigma_2$, if for all terms u and v such that $T \vdash \{u, v\}$ we have $u\sigma_1 =_E v\sigma_1 \Leftrightarrow u\sigma_2 =_E v\sigma_2$.*

Example 2. *Consider again the Otway-Rees protocol. As in Example 1, the initial knowledge of each principal is given by T_{a0}, T_{b0}, and T_{s0}, respectively. After receiving the first message, the knowledge of B becomes $T_{b1} = T_{b0} \cup \{x, y\} = \{A, B, S, N_b, K_{bs}, x, y\}$, where x and y denote M and $\{N_a \cdot M \cdot A \cdot B\}_{K_{as}}$, respectively.*

Although it appears to be a black box to B due to the lack of decryption key K_{as}, the message $\{N_a \cdot M \cdot A \cdot B\}_{K_{as}}$ cannot be interpreted as an arbitrary message. To see this, we let $\sigma_3 = [N_a \cdot N_a/y]$, $u =_s \mathtt{fst}(y)$, and $v =_s \mathtt{snd}(y)$. Note that $T_{b1} \vdash \{u, v\}$, $u\sigma_3 =_{E_{dy}} v\sigma_3 =_{E_{dy}} N_a$, and

$$u\sigma_1 =_s \mathtt{fst}(\{N_a \cdot M \cdot A \cdot B\}_{K_{as}})$$
$$v\sigma_1 =_s \mathtt{snd}(\{N_a \cdot M \cdot A \cdot B\}_{K_{as}})$$

Clearly, $u\sigma_1 \neq_{E_{dy}} v\sigma_1$ and $u\sigma_3 =_{E_{dy}} v\sigma_3$. Thus, $\sigma_1 \not\approx_{E_{dy},T_{b1}} \sigma_3$ follows immediately from Definition 3.2.

3.3 Recognizability

Our work is built upon the concept of *recognizablity* [23], which is proposed to formalize the idea of "verifying a message". Roughly speaking, a principal "recognizes" a message if he or she has certain expectation about its binary representation, or in other words, the message has an unambiguous interpretation. For instance, if Alice knows N_a and $\{N_a\}_{K_b^+}$, she is able to "recognize" K_b^+, because she can construct $\{N_a\}_{K_b^+}$ and compare it with her current knowledge.

Definition 3.3 (E-solver)**.** *Let $\vec{T} = \langle T, \sigma_0\rangle$ be a markup term set and let $X = fv(T)$. We say that substitution θ is an E-solver for $\vec{T}$ iff the following conditions hold*

(i). $\theta \approx_{E,T} \sigma_0$ and

(ii). if $\sigma \approx_{E,T} \sigma_0$ and $\sigma \preceq^X_E \theta$, then $\sigma =^X_E \theta$.

We define a minimum complete set of E-solvers (MCS) Θ *for $\vec{T}$ and write $\vec{T} \rightsquigarrow_E \Theta$ iff the following condition holds: σ is an E-solver of $\vec{T}$ iff there exists one and only one $\theta \in \Theta$ such that $\theta =^X_E \sigma$.*

Intuitively, an E-solver for $\vec{T}$ is a "most general" substitution that satisfies the operational equivalence imposed by $\vec{T}$. Since we are using relation $\preceq_E^X$ to characterize "generality", the "most general" one may not be unique (modulo E) up to renaming.

Definition 3.4 (Recognizability). *Let $\vec{T} = \langle T, \sigma_0 \rangle$ be a markup term set and t be a ground term. We say that t is* recognized as t' *by $\vec{T}$ under equational theory E if there exists an E-solver θ for $\langle T \cup \{x\}, \sigma_0 \circ [t/x] \rangle$ such that $x\theta =_E t'$, where x is a fresh variable. Moreover, we say that t is* recognizable *by $\vec{T}$ under equational theory E and write $\vec{T} \rhd_E t$ if t is* recognized as *itself by $\vec{T}$ under E.*

At this point, we can use recognizability to define the interpretation(s) of an incoming message. Let $\vec{T}$ denote a principal's knowledge. An incoming message t is interpreted as t' if and only if t is recognized as t' by $\vec{T}$ under E.

Example 3. *Let us consider the following ASW protocol, which is proposed by Asokan et. al. [6] for fair exchange and contract signing.*

1. $A \to B : \{K_a^+, K_b^+, M, hash(N_a)\}_{K_a^-}$
2. $B \to A : \{\{K_a^+, K_b^+, M, hash(N_a)\}_{K_a^-}, hash(N_b)\}_{K_b^-}$
3. $A \to B : N_a$
4. $B \to A : N_b$

We assume that the initial knowledge of A and B as follows.

$$T_{a0} = \{M, A, B, K_a^+, K_b^+, K_a^-, N_a\}$$
$$T_{b0} = \{A, B, K_a^+, K_b^+, K_b^-, N_b\}$$

Let σ_{a0} and σ_{b0} be the intended interpretations of the messages received by A and B, respectively. After the protocol run is completed, the knowledge of each principal becomes

$$\vec{T_a} = \langle T_{a0} \cup \{x_2, x_4\}, \sigma_{a0} \rangle$$
$$\vec{T_b} = \langle T_{b0} \cup \{x_1, x_3\}, \sigma_{b0} \rangle$$

where x_1 to x_4 signify the four incoming messages, and

$$\sigma_{a0} = [\{\{K_a^+ \cdot K_b^+ \cdot M \cdot \mathtt{hash}(N_a)\}_{K_a^-} \cdot \mathtt{hash}(N_b)\}_{K_b^-}/x_2, N_b/x_4]$$
$$\sigma_{b0} = [\{K_a^+ \cdot K_b^+ \cdot M \cdot \mathtt{hash}(N_a)\}_{K_a^-}/x_1, N_a/x_3]$$

Let
$$u_1 =_s \mathtt{fst}(\mathtt{pdec}(x_2, K_b^+))$$
$$u_2 =_s \{K_a^+ \cdot K_b^+ \cdot M \cdot \mathtt{hash}(N_a)\}_{K_a^-}$$
$$u_3 =_s \mathtt{snd}(\mathtt{pdec}(x_2, K_b^+))$$
$$u_4 =_s \mathtt{hash}(x_4)$$

Then, from A's point of view, $u_1\sigma_{a0} =_{E_{dy}} u_2\sigma_{a0}$ and $u_3\sigma_{a0} =_{E_{dy}} u_4\sigma_{a0}$. Note that A knows $u_1, \cdots, u_4$ and $\sigma_{a0} \approx_{E_{dy}, T_{a0} \cup \{x_2, x_4\}} \sigma_a$.

Let σ_a and σ_b be possible interpretations of ambiguous messages received by A and B, respectively. By operational equivalence, we have $u_1\sigma_a =_{E_{dy}} u_2\sigma_a$ and $u_3\sigma_a =_{E_{dy}} u_4\sigma_a$, which hold iff

$$x_2\sigma_a =_{E_{dy}} \{\{K_a^+ \cdot K_b^+ \cdot M \cdot \mathtt{hash}(N_a)\}_{K_a^-} \cdot \mathtt{hash}(x_4)\sigma_a\}_{K_b^-}$$

Now, it is not hard to see that substitution

$$\theta_a = [\{\{K_a^+ \cdot K_b^+ \cdot M \cdot \mathtt{hash}(N_a)\}_{K_a^-} \cdot \mathtt{hash}(x_4)\}_{K_b^-}/x_2]$$

is an E_{dy}-solver for $\vec{T_a}$. In fact, θ_a is the only E_{dy}-solver for $\vec{T_a}$ up to variable renaming and term rewriting. So, the two messages received by A should be interpreted as $\{\{K_a^+ \cdot K_b^+ \cdot M \cdot \mathtt{hash}(N_a)\}_{K_a^-} \cdot \mathtt{hash}(x_4)\}_{K_b^-}$ and x_4, respectively.

A similar analysis shows that substitution

$$\theta_b = [\{K_a^+ \cdot K_b^+ \cdot y \cdot \mathtt{hash}(x_3)\}_{K_a^-}/x_1]$$

is the only E_{dy}-solver for $\vec{T_a}$ up to variable renaming and term rewriting. So, the two messages received by B should be interpreted as $\{K_a^+ \cdot K_b^+ \cdot y \cdot \mathtt{hash}(x_3)\}_{K_a^-}$ and x_3, respectively.

Now, we discuss how to obtain a MCS for a given markup term set. To determine E-solvers, we first construct conditions imposed by operational equivalence, such as $u_1\sigma_{a0} =_{E_{dy}} u_2\sigma_{a0}$ and $u_3\sigma_{a0} =_{E_{dy}} u_4\sigma_{a0}$ in the previous example, and then update substitutions by solving those equations. This is reminiscent of the constraint solving approach proposed by Millen and Shmatikov [29]. Here, we apply the constraint solving methodology to find a MCS.

A *constraint* of a markup term set $\langle T, \sigma \rangle$ under equational theory E is an unordered pair (u, v) of terms such that $T \vdash \{u, v\}$, $u\sigma =_E v\sigma$, and $u \neq_E v$. We say that θ is an *E-unifier* of a constraint set $\mathcal{C}$ and write $\theta \vDash_E \mathcal{C}$ if $u\theta =_E v\theta$ for every $(u, v) \in \mathcal{C}$. Substitution set Θ is a *minimal complete set of E-unifier* (MCU) of $\mathcal{C}$, written as $\mathcal{C} \rightsquigarrow_E \Theta$, if the following conditions hold:

- $\theta \vDash_E \mathcal{C}$ for each $\theta \in \Theta$,
- there exists a $\theta \in \Theta$ such that $\theta \preceq_E^X \sigma$ whenever $\sigma \vDash_E \mathcal{C}$,
- two distinct elements of Θ are incomparable w.r.t. $\preceq_E^X$.

Definition 3.5 (Constraint Base). *Let $\vec{T}$ be a markup term set and E an equational theory. Suppose that $\mathcal{C}$ is the set of all constraints of $\vec{T}$ under E and $\mathcal{C} \rightsquigarrow_E \Theta$. Then, we say that $\mathcal{C}'$ is a* constraint base *of $\vec{T}$ under E if $\mathcal{C}'$ is the smallest constraint set satisfying that $\mathcal{C}' \rightsquigarrow_E \Theta$ and $\mathcal{C}'$ is finite.*

This is analogous to the definition "finite basis property" given in [13]. In Example 3, we see $\{(u_1, u_2), (u_3, u_4)\}$ is a constraint base of $\vec{T_a}$.

Proposition 3.6. *Let $\vec{T} = \langle T, \sigma \rangle$ be a markup term set. Suppose that $\mathcal{C}$ is a constraint base of $\vec{T}$. Then, $\vec{T} \rightsquigarrow_E \Theta$ iff $\mathcal{C} \rightsquigarrow_E \Theta$.*

In view of Proposition 3.6, we reduce the problem of obtaining a MCS to that of finding and solving a constraint base. This problem is undecidable in general, because E-unification is undecidable [33, Chapter 8]. Nonetheless, restricting ourselves to some specific equational theories is likely to yield decidable results. Notably, a procedure is given in [24] to decide recognizability under the standard Dolev-Yao model. Due to space limit, we do not pursue these further here. Henceforth, let us assume that constraint bases are obtained.

4. THE IDEAL SEMANTICS

Having discussed the interpretation(s) of a message, we now discuss how to extract ideal semantics from protocol narrations. We avoid introducing new formalism and base the semantics on strand space model [20], a widely-used formalism in modeling and verifying security protocols [22, 34, 29]. In this paper, strands serve three purposes: (a) describing a real protocol execution trace; (b) providing protocol semantics; and (c) specifying a protocol implementation (in the next section).

4.1 Strands

In the strand space model, an *event* is a signed term $+t$ or $-t$ that indicates the *sending* (+) or *receiving* (-) of a message. A *strand* $\vec{s}$ is a finite sequence of nodes that describe the events happening at a legitimate party or an intruder; the i-th node of the strand is denoted by $\vec{s}[i]$. Nodes within the same strand and among different strands are linked by the relationships $\Rightarrow$ and $\rightarrow$, respectively. More specifically, $\Rightarrow$ is used to indicate a protocol role's execution sequence; and $\rightarrow$ is used to specify the communication between different principals. A *bundle* is a finite subgraph of strand spaces that can be viewed as a snapshot of a protocol execution. Figure 2 shows a bundle that illustrates the expected execution of the ASW protocol.

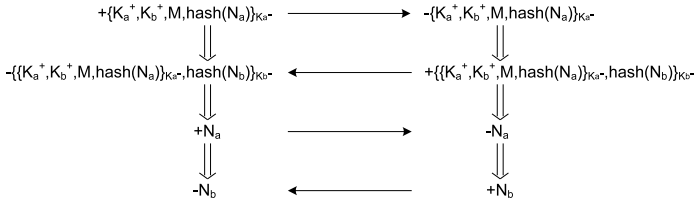


Figure 2: ASW protocol: a bundle.

Each strand in a bundle describing an expected protocol execution is associated with a role of the protocol. For instance, the two strands in Figure 2 correspond to the roles A and B in the ASW protocol. We have seen that messages exchanged between principals (taking some roles) can be interpreted considerably differently; and an unrecognizable (part of) message is often treated as a free variable. For example, role A in the ASW protocol should be specified by

$$\begin{aligned}&\underline{A[M, A, B, N_a, x]}\\ &\langle + \{K_a^+ \cdot K_b^+ \cdot M \cdot \mathtt{hash}(N_a)\}_{K_a^-},\\ &\quad - \{\{K_a^+ \cdot K_b^+ \cdot M \cdot \mathtt{hash}(N_a)\}_{K_a^-}, \mathtt{hash}(x)\}_{K_b^-},\\ &\quad + N_a, -x\rangle\end{aligned}$$

where x is instantiated to N_b in a normal protocol run.

We associate strand $\vec{s}$ with a ground term set $\vec{s}[0]$ to describe its initial knowledge, and use $\mathcal{K}_i(\vec{s})$ to denote the knowledge of a principal (at step i) taking the role specified by $\vec{s}$. That is,

$$\mathcal{K}_i(\vec{s}) = \begin{cases} \vec{s}[0] & \text{if } i = 0\\ \mathcal{K}_{i-1}(\vec{s}) \cup \{t\} & \text{if } i > 0 \text{ and } \vec{s}[i] = -t\\ \mathcal{K}_{i-1}(\vec{s}) & \text{otherwise}\end{cases}$$

To account for ambiguous messages, we inductively define $\vec{\mathcal{K}}_i(\vec{s})$ as follows

$$\vec{\mathcal{K}}_i(\vec{s}) = \begin{cases} \langle \vec{s}[0], [] \rangle & \text{if } i = 0\\ \langle \vec{\mathcal{K}}_{i-1}(\vec{s}) \downarrow_{ts} \cup \{x\}, \vec{\mathcal{K}}_{i-1}(\vec{s}) \downarrow_{subs} \circ [t/x] \rangle & \\ \quad \text{where } x \text{ is a fresh variable, if } i > 0 \text{ and } \vec{s}[i] = -t & \\ \vec{\mathcal{K}}_{i-1}(\vec{s}) & \text{otherwise}\end{cases}$$

The subscript i will be omitted if $i = length(\vec{s})$.

4.2 Execution Traces

In this subsection, we use execution traces to describe real protocol executions and formalize the meaning of "a protocol execution is in compliance with the protocol narration".

An *execution trace* or simply a *trace tr* is a strand containing no variable (i.e., ground strand). Clearly, every protocol execution can be described by a set of execution traces. It is natural to parse a protocol narration into a set of traces; we will always assume that such traces are obtained, and refer to those traces as *narrative traces*.

We say that two strands $\vec{s}_1$ and $\vec{s}_2$ are *isomorphic* iff $\vec{\mathcal{K}}(\vec{s}_1) \downarrow_{ts}$ and $\vec{\mathcal{K}}(\vec{s}_2) \downarrow_{ts}$ are identical up to variable renaming, that is, there exists a variable renaming substitution η that $\vec{\mathcal{K}}(\vec{s}_1) \downarrow_{ts} \eta = \vec{\mathcal{K}}(\vec{s}_2) \downarrow_{ts}$. For simplicity, we assume that $\vec{\mathcal{K}}(\vec{s}_1) \downarrow_{ts} = \vec{\mathcal{K}}(\vec{s}_2) \downarrow_{ts}$ whenever they are isomorphic. We say that $\vec{s}_1$ and $\vec{s}_2$ are *operationally equivalent* in equational theory E, written as $\vec{s}_1 \approx_E \vec{s}_2$, iff $\vec{\mathcal{K}}(\vec{s}_1) \downarrow_{subs} \approx_{E,T} \vec{\mathcal{K}}(\vec{s}_2) \downarrow_{subs}$ where $T = \vec{\mathcal{K}}(\vec{s}_1) \downarrow_{ts} = \vec{\mathcal{K}}(\vec{s}_2) \downarrow_{ts}$.

Definition 4.1. *Given an equational theory E, we say that an execution trace tr is* in compliance with *a set of strands $\vec{S}$, written as $\vec{S} \rightsquigarrow_E tr$, iff $tr \approx_E \vec{s}$ for some $\vec{s} \in \vec{S}$. Two sets of strands $\vec{S}_1$ and $\vec{S}_2$ are* equivalent*, written $\vec{S}_1 \approx_E \vec{S}_2$, if all, and only, execution traces in compliance with $\vec{S}_1$ are in compliance with $\vec{S}_2$.*

4.3 Semantics

To obtain an ideal semantics of a protocol narration, it is essential to capture all possible execution traces that are in compliance with the narration.

Definition 4.2 (Ideal Semantics)**.** *Let $\vec{S}$ be a set of strands and TR_0 be a set of narrative traces. Given an equational theory E, we say that $\vec{S}$ is an* ideal semantics *of TR_0 iff $\vec{S} \approx_E TR_0$.*

Unfortunately, there is often an infinite number of execution traces that are in compliance with the set of narrative traces TR_0. So, it is preferable to use "patterns" to capture those execution traces thanks to fully fledged interpretations of incoming messages. For example, in an arbitrary successful run of the Otway-Reese protocol the last message should look like $\{N_a, x\}_{K_{as}}$, because K_{ab} is recognized as ϵ and is thus replaced by a free variable x. This approach resembles the "pattern-matching" technique widely-used in formal protocol analysis [34, 10, 16, 7].

Our definition of "recognized as" (Definition 3.4) fits the intuitive understanding of "patterns". Given a narrative trace tr_0, we can use the MCS of $\vec{\mathcal{K}}(tr_0)$ to characterize all possible incoming messages in a succesful protocol run.

Altogether, we obtain Algorithm 1 to extract an ideal semantics from a protocol narration. The algorithm takes an input set of narrative traces TR_0 and an equational theory E, and produces an ideal semantics of TR_0.

The main loop of the algorithm selects an arbitrary narrative trace tr and obtain a set of operationally equivalent strands. It has two stages. In the first stage, from line 3 to line 9, it construct an abstract strand by replacing each incoming message with a fresh variable and replacing each outgoing message with its corresponding recipe. In the second stage, it first computes a MCS Θ of $\vec{\mathcal{K}}(tr)$ in line 10. We see that each $\theta \in \Theta$ corresponds to an interpretation of the incoming messages, because, by Definition 3.3, it is operationally equivalent to $\vec{\mathcal{K}}(tr)$ and is in its most general

Algorithm 1 **Extract-Ideal-Semantics**

Input: a set of narrative traces TR_0, equational theory E
Output: a set of strands $\vec{S}$

1: $\vec{S} \leftarrow \emptyset$
2: **for** each $tr_0 \in TR_0$
3: $\vec{s}_p \leftarrow \langle\rangle$, $\mathbb{S} \leftarrow \emptyset$
/* *specify initial knowledge* */
4: append strand $\vec{s}_p$ with $tr_0[0]$
/* *obtain a markup term set representing the principal's knowledge upon protocol completion* */
5: **for** $j = 1$ to $length(tr_0)$
6: **if** $tr_0[j] = +t$ for some term t **then**
7: append strand $\vec{s}_p$ with node $+t'$ where t' is a recipe of t
8: **if** $tr_0[j] = -t$ for some term t **then**
9: append strand $\vec{s}_p$ with node $-x$ where x is a fresh variable
10: obtain a MCS Θ of $\vec{\mathcal{K}}(tr_0)$
11: $\mathbb{S} \leftarrow \mathbb{S} \cup \{\vec{s}_p\theta\}$ for each $\theta \in \Theta$
12: $\vec{S} \leftarrow \vec{S} \cup \mathbb{S}$
13: **return** $\vec{S}$

form. So, in line 11, we include all strands associated with those interpretations in output ideal semantics.

Theorem 4.3. *Let TR_0 be a set of narrative traces. Then, **Extract-Ideal-Semantics**(TR_0, E) returns an ideal semantics of TR_0.*

Proof. Let $\vec{S}_I$ = **Extract-Ideal-Semantics**(TR_0, E). It suffices to show that $\vec{S}_I \approx_E TR_0$. That is, an arbitrary execution trace tr is in compliance with $\vec{S}_I$ if and only if it is in compliance with TR_0.

("If" part) By $TR_0 \rightsquigarrow_E tr$, there exists a trace $tr_0 \in TR_0$ such that $tr \approx_E tr_0$. That is, $\vec{\mathcal{K}}(tr) \downarrow_{subs} \approx_{E,T} \vec{\mathcal{K}}(tr_0) \downarrow_{subs}$ where $T = \vec{\mathcal{K}}(tr) \downarrow_{ts} = \vec{\mathcal{K}}(tr_0) \downarrow_{ts}$. By Definition 3.3, there exists a $\theta \in \Theta$ such that $\theta \preceq_E^X \vec{\mathcal{K}}(tr) \downarrow_{subs}$ and $\theta \approx_{E,T} \vec{\mathcal{K}}(tr_0) \downarrow_{subs}$, where Θ is a MCS of $\vec{\mathcal{K}}(tr_0)$ and $X = fv(T)$. We note from Algorithm 1 that $\vec{\mathcal{K}}(\vec{s}_p\theta) \downarrow_{ts} = T$ and $\vec{\mathcal{K}}(\vec{s}_p\theta) \downarrow_{subs} = \theta$. So, $tr \approx_E \vec{s}_p\theta \in \vec{S}$, that is, $\vec{S}_I \rightsquigarrow_E tr$.

("Only if" part) By $\vec{S}_I \rightsquigarrow_E tr$, we see from Algorithm 1 that there exists a strand $\vec{s}_p\theta \in \vec{S}$ such that $tr \approx_E \vec{s}_p\theta$. That is, $\vec{\mathcal{K}}(tr) \downarrow_{subs} \approx_{E,T} \vec{\mathcal{K}}(\vec{s}_p\theta) \downarrow_{subs} = \theta$ where $T = \vec{\mathcal{K}}(tr) \downarrow_{ts} = \vec{\mathcal{K}}(\vec{s}_p\theta) \downarrow_{ts}$. On the other hand, we notice that there exists a trace $tr_0 \in TR_0$ such that $\vec{\mathcal{K}}(tr_0) \downarrow_{ts} = \vec{\mathcal{K}}(\vec{s}_p\theta) \downarrow_{ts}$. Besides, since θ is an E-solver of $\vec{\mathcal{K}}(tr_0)$, we have $\vec{\mathcal{K}}(tr_0) \downarrow_{subs} \approx_{E,T} \theta$. Consequently, we obtain $tr \approx_E tr_0$ for some $tr_0 \in TR_0$ and thus $TR_0 \rightsquigarrow_E tr$. □

We stress that a protocol could be executed in a hostile environment. A principal may intentionally abort a protocol before completion. So, in Algorithm 1 the narrative traces must include all partial protocol runs [15]. To highlight the effect of partial runs on the ideal semantics, let us consider an example.

Example 4. *We consider the following contrived protocol:*

1. $A \rightarrow B : M_1$
2. $B \rightarrow A : M_2$
3. $A \rightarrow B : M_3$
4. $B \rightarrow A : M_4$

We assume that the initial knowledge of A and B as follows.

$$T_{a0} = \{M_1, M_3\}$$
$$T_{b0} = \{M_2, M_4, \{M_1\}_{M_3}\}$$

The narrative trace of role B is

$$\vec{s}_1 = \langle\{M_2, M_4, \{M_1\}_{M_3}\}, -M_1, +M_2, -M_3, +M_4\rangle$$

It is not hard to see that another possible partial run is

$$\vec{s}_2 = \langle\{M_2, M_4, \{M_1\}_{M_3}\}, -M_1, +M_2\rangle$$

At first, for both strands we get

$$\vec{\mathcal{K}}_4(\vec{s}_1) = \langle\{M_2, M_4, \{M_1\}_{M_3}, x_1, x_3\}, [M_1/x_1, M_3/x_3]\rangle$$
$$\vec{\mathcal{K}}_2(\vec{s}_2) = \langle\{M_2, M_4, \{M_1\}_{M_3}, x_1\}, [M_1/x_1]\rangle$$

Let Θ_1 and Θ_2 be the MCS for $\vec{\mathcal{K}}_4(\vec{s}_1)$ and $\vec{\mathcal{K}}_2(\vec{s}_2)$, respectively. Note that $\{x_1\}_{x_3}[M_1/x_1, M_3/x_3] =_{E_{dy}} \{M_1\}_{M_3}$. Then, it can be shown that

$$\Theta_1 = \{[M_1/x_1, M_3/x_3]\},\ \Theta_2 = \{[]\}$$

Thus, in a normal protocol run the first and third messages are interpreted as M_1 and M_3, respectively, whereas in a partial protocol run the first message is interpreted as free variable x_1. That is to say, if the protocol execution succeeds, B only accepts M_1 as the first message, otherwise any message will be accepted.

For now, it is not hard to see the ideal semantics (of role B) contains the following two strands:

$$\vec{s}_1' = \{M_2, M_4, \{M_1\}_{M_3}\}, -M_1, +M_2, -M_3, +M_4\rangle$$
$$\vec{s}_2' = \{M_2, M_4, \{M_1\}_{M_3}\}, -x_1, +M_2\rangle$$

5. FROM IDEAL IMPLEMENTATION TO REFINED IMPLEMENTATION

In this section, we turn our attention to protocol implementations. First, we extend the definition of a strand to allow for specifying internal actions. Next, we define an ideal implementation according to the ideal semantics of a protocol. Since the ideal implementation may not exist, we then use prudent and refined implementations to approximate it.

Unlike the ideal semantics where messages are regarded as symbolic expressions, in real protocol implementation every message is merely a bit string which has potentially ambiguous interpretations. That's why an ideal semantics highlights external patterns of an incoming message, whereas an implementation emphasizes the internal actions of protocol participants. Initially, in a protocol implementation, every incoming message is ambiguous and thus should be indicated by a fresh variable. Only after performing some condition checks on messages, the recipient would gain some certainty. For example, in the ASW protocol (see Example 3) A ought to check whether $\mathtt{fst}(\mathtt{pdec}(x_2, K_b^+))$ equals to the first sent message, where x_2 signifies the received message.

To specify internal actions, we define a *check* event as $check(u = v)$ or $check(u \neq v)$, where both u and v are terms. We will use "equality check" and "inequality check" to discriminate them. An *implementation strand* $\vec{p}$ is a strand that allows check events, and all *receive* events contain only free variables that are pairwise distinct. We say that an implementation strand $\vec{p}$ is *feasible* under equational theory E iff the following conditions hold:

(i). $\mathcal{K}_i(\vec{p}) \vdash_E t$ whenever $\vec{p}[i] = +t$, and

(ii). $\mathcal{K}_i(\vec{p}) \vdash_E \{u, v\}$ whenever $\vec{p}[i]$ is $check(u = v)$ or $check(u \neq v)$.

This coincides with the definitions of *executability* and *feasibility* in [10].

Since an implementation strand makes internal checks explicit, it can be easily mapped to a practical implementation. For this reason, we define *protocol implementation* $\mathcal{P}$ as a set of implementation strands; each corresponds to a role of the protocol. For convenience, we use $\vec{p}\downarrow$ to denote a strand obtained from $\vec{p}$ by removing all nodes representing *check* events.

Definition 5.1 (In Compliance with)**.** *An execution trace tr is* in compliance with *a protocol implementation $\mathcal{P}$ iff there exists an implementation $\vec{p} \in \mathcal{P}$ and a substitution θ such that $tr = \vec{p}\downarrow\theta$ and for each $check(u = v)$ (resp. $check(u \neq v)$) event in $\vec{p}$ we have $u\theta =_E v\theta$ (resp. $u\theta \neq_E v\theta$).*

Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be two protocol implementations. We say that $\mathcal{P}_1$ encompasses *$\mathcal{P}_2$, and write $\mathcal{P}_1 \subseteq_E \mathcal{P}_2$, if all execution traces in compliance with $\mathcal{P}_2$ are also in compliance with $\mathcal{P}_1$; and $\mathcal{P}_1$ and $\mathcal{P}_2$ are* equivalent*, written $\mathcal{P}_1 \approx_E \mathcal{P}_2$, iff $\mathcal{P}_1 \subseteq_E \mathcal{P}_2$ and vice versa. As usual, we write $\mathcal{P}_1 \subset_E \mathcal{P}_2$ for $\mathcal{P}_1 \subseteq_E \mathcal{P}_2$ and $\mathcal{P}_1 \not\approx_E \mathcal{P}_2$. These notations are extended in the obvious way to sets of strands.*

5.1 Ideal Implementation

Definition 5.2 (Ideal Implementation)**.** *Let $\vec{S}$ be an ideal protocol semantics. An* ideal implementation *of $\vec{S}$ is defined as a protocol implementation $\mathcal{P}$ such that $\mathcal{P} \approx_E \vec{S}$.*

Theorem 5.3. *Let $\vec{S}$ be an ideal protocol semantics of protocol narration TR_0. The ideal implementation of $\vec{S}$ exists if and only if $\vec{S}$ does not contain any free variable.*

Proof. (Sketch) ("If" part) As we will see in the next subsection, Algorithm 2 gives an implementation $\mathcal{P}$. To prove $\mathcal{P} \approx_E \vec{S}$, by Definition 4.2 it suffices to show that $\mathcal{P} \approx_E TR_0$. That is, $\mathcal{P} \rightsquigarrow_E tr \Leftrightarrow \vec{S} \rightsquigarrow_E tr$.

We begin with the "$\Rightarrow$" direction. By $\mathcal{P} \rightsquigarrow_E tr$, we have $tr = \vec{p}\downarrow\sigma$ for some implementation $\vec{p}$ and substitution σ. Let $\mathcal{C}$ be the set of constraints checked in $\vec{p}$ and $\mathcal{C} \rightsquigarrow_E \Theta$. We see from Definition 5.1 that $\theta \preceq_E^X \sigma$ for some $\theta \in \Theta$ and $X = fv(\vec{\mathcal{K}}(tr)\downarrow_{ts})$. Notice that there exists a narrative trace $tr_0 \in TR_0$ such that $\mathcal{C}$ is a constraint base of $\vec{\mathcal{K}}(tr_0)$. It follows from Proposition 3.6 that $\vec{\mathcal{K}}(tr_0) \rightsquigarrow_E \Theta$. By Definition 3.3, we get $\theta \approx_{E,T} \mathcal{K}(\vec{tr}_0)\downarrow_{subs}$. Moreover, since $\vec{S}$ does not contain any free variable, we know Θ contains only ground substitutions and thus $\sigma =_E^X \theta$. Consider now $\vec{\mathcal{K}}(tr)\downarrow_{ts} = \vec{\mathcal{K}}(tr_0)\downarrow_{ts} = T$ and $\vec{\mathcal{K}}(tr)\downarrow_{subs} = \sigma =_E^X \theta \approx_{E,T} \vec{\mathcal{K}}(tr_0)\downarrow_{subs}$, we have $tr \approx_E tr_0$ and thus $TR_0 \rightsquigarrow_E tr$. The reverse direction can be shown in a similar way.

("Only if" part) We will show that if $\vec{S}$ contains free variable(s), then the ideal implementation does not exist. The main reason is that, when an ideal semantics contains free variable(s), it is impossible to use even an infinite set of equality and/or inequality checks to establish operational equivalence.

For equality check, we note that constraints are implied by operational equivalence $\sigma_0 \approx_{E,T} \sigma$. They, however, do not suffice to characterize operational equivalence. In other words, we cannot base operational equivalence on a possibly infinite set of equations. Here is an example to show why. Let $T = \{N_b, x\}$ and $\sigma_0 = [\{N_b\}_{K_{as}}/x]$, and suppose that $\sigma_0 \approx_{E_{dy},T} \sigma$. It is clear that there is no constraint of $\langle T, \sigma_0\rangle$. However, it does not follow that $\sigma_0 \approx_{E_{dy},T} \sigma$ holds for an arbitrary substitution σ. For instance, by letting $\sigma = [N_c \cdot N_c/x]$, we get $\mathtt{fst}(x)\sigma =_{E_{dy}} \mathtt{snd}(x)\sigma$ and $\mathtt{fst}(x)\sigma_0 \neq_{E_{dy}} \mathtt{snd}(x)\sigma_0$. So, $\sigma_0 \not\approx_{E_{dy}} \sigma$.

Incorporating inequality checks may not help either. As an example, let us we consider a substitution σ that satisfies $\sigma \approx_{E_{dy},\{N_a,K_a^+,x\}} [N_b/x]$. To establish the operational equivalence, we have to check $x\sigma \neq_{E_{dy}} t\sigma$ for every term t such that $\{N_a, K_a^+, x\} \vdash t$. □

5.2 Coarse and Prudent Implementations

A *coarse implementation* of an ideal protocol semantics $\vec{S}$ is a protocol implementation $\mathcal{P}$ such that $\vec{S} \subseteq_E \mathcal{P}$.

Definition 5.4 (Prudent Implementation)**.** *Given an ideal protocol semantics $\vec{S}$, we define a* prudent implementation *of $\vec{S}$ as a protocol implementation $\mathcal{P}$ such that*

(i). $\vec{S} \subseteq_E \mathcal{P}$;

(ii). $\mathcal{P}$ does not contain any inequality check event;

(iii). there does not exist an implementation $\mathcal{P}'$ that satisfies (i), (ii), and $\mathcal{P}' \subset_E \mathcal{P}$.

Making Checks Explicit. As we have seen, the constraint base maximizes the chance to check non-trivial equalities implied by a protocol narration. It can be used to construct check events in strands. Suppose that $\mathcal{C}$ is a constraint base of markup term set $\vec{T}$, which models a principal's knowledge after completing a protocol. Then, whenever possible, the principal should check each constraint (u, v) in a constraint base and abort upon constraint violation (i.e., $u\sigma \neq_E v\sigma$). Note that a principal might not be able to check those constrains all at once. Let $\vec{T}_i = \langle \mathcal{K}_i, \sigma_i\rangle$ be a principal's knowledge after the i-th step of a protocol. Then, he can check a constraint (u, v) whenever $\mathcal{K}_i \vdash \{u, v\}$.

For example, at step 2 of the ASW protocol, Alice is able to check constraint (u_1, u_2) but not (u_3, u_4), which becomes checkable only after she receives the last message. So, the strand of role A becomes:

$$\underline{A[M, A, B, N_a, x_2, x_4]}$$
$$\langle \{M, A, B, K_a^+, K_b^+, K_a^-, N_a\},$$
$$+\{K_a^+ \cdot K_b^+ \cdot M \cdot \mathtt{hash}(N_a)\}_{K_a^-},\ -x_2,$$
$$check(\mathtt{fst}(\mathtt{pdec}(x_2, K_b^+)) = \{K_a^+ \cdot K_b^+ \cdot M \cdot \mathtt{hash}(N_a)\}_{K_a^-}),$$
$$+N_a,\ -x_4,\ check(\mathtt{snd}(\mathtt{pdec}(x_2, K_b^+)) = \mathtt{hash}(x_4))\rangle$$

Interpreting Outgoing Messages. The above example of the ASW protocol is too restrictive, because both terms in the *send* events are deducible from the principal's initial knowledge and thus avoid dealing with outgoing messages, which is not always the case. For instance, the third message (i.e., $M \cdot \{N_a \cdot K_{ab}\}_{K_{as}} \cdot \{N_b \cdot K_{ab}\}_{K_{bs}}$) in the Otway-Reese protocol, which contains nonces generated by A and B, is obviously not deducible from S. Consequently, we need to be clear on the interpretation of outgoing messages as well when specifying the implementation.

Although strands are assumed to be well-formed, how to generate the outgoing messages is unspecified. To see this, let us consider a narrative trace $\vec{s}$. Without loss of generality, assume that $\vec{s}[i] = +t$ and $\vec{\mathcal{K}}_i(\vec{s}) = \langle T_i, \sigma_i \rangle$. The meaning of well-formedness is twofold. First, we get $\mathcal{K}_i(\vec{s}) \vdash_E t$ in terms of the original narrative trace $\vec{s}$. Second, we should also achieve $T_i \vdash t'$ and $t'\sigma_i =_E t$ in the new compiled strand. This accords with Proposition 2.1, as $T_i\sigma_i = \mathcal{K}_i(\vec{s})$, and t' is a recipe of t.

The key to our interpretation is therefore to find a recipe for each outgoing message. Unfortunately, the recipe may not be unique, posing a major hurdle in interpreting an outgoing message.

Example 5. *To make this more concrete, let us consider a very simple protocol.*

$$1.\ A \to B : \{K_{ab}\}_{K_b^+}$$
$$2.\ B \to A : \{M\}_{K_{ab}}$$

Suppose that the initial knowledge of B is $T_{b0} = \{A, B, M, K_a^+, K_b^+, K_b^-, K_{ab}\}$. The narrative trace of role B is $\vec{s} = \langle T_{b0}, -\{K_{ab}\}_{K_b^+}, +\{M\}_{K_{ab}} \rangle$ Then, $\mathcal{K}_2(\vec{s}) = T_{b0} \cup \{\{M\}_{K_{ab}}\}$ and $\vec{\mathcal{K}}_2(\vec{s}) = \langle T_{b0} \cup \{x_1\}, [\{K_{ab}\}_{K_b^+}/x_1] \rangle$. By letting $t_1' =_s \{M\}_{K_{ab}}$ and $t_2' =_s \mathtt{penc}(M, \mathtt{pdec}(x_1, K_b^-))$, we get $T_{b0} \cup \{x_1\} \vdash \{t_1', t_2'\}$ and

$$t_1'[\{K_{ab}\}_{K_b^+}/x_1] =_{E_{dy}} t_2'[\{K_{ab}\}_{K_b^+}/x_1] =_{E_{dy}} \{M\}_{K_{ab}}$$

Here, both t_1' and t_2' are recipes of $\{M\}_{K_{ab}}$, corresponding to two different ways of generating the message $\{M\}_{K_{ab}}$. If we admit t_1' as the recipe, then the compiled strand of role B is

$$\vec{s}_1 = \langle T_{b0}, -x_1, check(\mathtt{pdec}(x_1, K_b^-) = K_{ab}), + \{M\}_{K_{ab}} \rangle \quad (1)$$

Otherwise (t_2' as the recipe), the compiled strand becomes

$$\vec{s}_2 = \langle T_{b0}, -x_1, check(\mathtt{pdec}(x_1, K_b^-) = K_{ab}), + \mathtt{penc}(M, \mathtt{pdec}(x_1, K_b^-)) \rangle \quad (2)$$

Due to the check events, $\vec{s}_1$ and $\vec{s}_2$ are equivalent in a sense that no ambiguity arises from the choice of recipe. On the contrary, if we eliminate the check events, then the implementations defined by $\vec{s}_1$ and $\vec{s}_2$ differ significantly.

Thanks to the internal checks, we make the following claim, which allows us to choose any recipe of an outgoing message without affecting the result of the implementation.

Claim 5.5. *The prudent implementation remains invariant under different interpretations of outgoing messages.*

Incorporating the above considerations, we obtain the following algorithm to derive a prudent implementation from a set of narrative traces TR_0.

The algorithm creates an implementation strand for each narrative trace. The construction starts by using the narrative trace to compute a constraint base. For a node with *receive* event, from line 6 to line 9, it updates knowledge and construct a new equality check event whenever it becomes feasible. For a node with *send* event, from line 10 to line 11, the algorithm simply chooses an arbitrary recipe of the outgoing message due to Claim 5.5.

Algorithm 2 Derive-Prudent-Implementation

Input: a set of narrative traces TR_0, equational theory E
Output: a protocol implementation $\mathcal{P}$

1: $\vec{S} \leftarrow \emptyset$
2: **for** each narrative trace $tr_0 \in TR_0$
3: obtain a constraint base $\mathcal{C}$ of $\vec{\mathcal{K}}(tr_0)$ (under E)
/* *construct an implementation strand $\vec{p}$* */
4: $\vec{p} \leftarrow \langle tr_0[0] \rangle$
5: **for** $i = 1$ to $length(tr_0)$
/* *find all new constraints that are enabled by the incoming message* */
6: **if** $tr_0[i] = -t$ for some term t **then**
7: append strand $\vec{p}$ with node $-x_i$
8: **for each** $(u, v) \in \mathcal{C}$ such that $\mathcal{K}(\vec{p}) \vdash \{u, v\}$ and $\mathcal{K}_{l-1}(\vec{p}) \nvdash \{u, v\}$ where $l = length(\vec{p})$ **do**
9: append strand $\vec{p}$ with node $check(u, v)$
/* *choose an arbitrary recipe as an interpretation of the outgoing message* */
10: **if** $tr_0[i] = +t$ for some term t **then**
11: append strand $\vec{p}$ with node $+t'$ where t' is a recipe of t
12: $\vec{S} \leftarrow \vec{S} \cup \{\vec{p}\}$
13: **return** $\vec{S}$

Theorem 5.6. *Let TR_0 be a set of narrative traces and $\vec{S}$ be an ideal semantics of TR_0. Then, **Derive-Prudent-Implementation**(TR_0) returns an prudent implementation of $\vec{S}$.*

5.3 Refined Implementation

To illustrate the idea of implemention refinement, let us reexamine the motivating example given in Section 1. We recapitulate the well-known type flaw attack here.

$$1.\ A \to B \quad : \quad M, A, B, \{N_a, M, A, B\}_{K_{as}}$$
$$4.\ I(B) \to A \quad : \quad M, \{N_a, M, A, B\}_{K_{as}}$$

After initiating the first message, A is expecting from B the message $M \cdot \{N_a \cdot K_{ab}\}_{K_{as}}$, which is forged by an intruder I. The intruder I impersonates B and then replays an intercepted message to A. It is not hard to see that the narrative trace for role A is

$$\begin{aligned} tr_A = & \underline{A[M, A, B, S, N_a, K_{as}, K_{ab}]} \\ & \langle \{M, A, B, S, N_a, K_{as}\}, \\ & \quad + M \cdot A \cdot B \cdot \{N_a \cdot M \cdot A \cdot B\}_{K_{as}}, \\ & \quad - \{N_a \cdot K_{ab}\}_{K_{as}} \rangle \end{aligned}$$

Likewise, we get narrative trace tr_I describing the attack scenario.

$$\begin{aligned} tr_I = & \underline{A[M, A, B, S, N_a, K_{as}]} \\ & \langle \{M, A, B, S, N_a, K_{as}\}, \\ & \quad + M \cdot A \cdot B \cdot \{N_a \cdot M \cdot A \cdot B\}_{K_{as}}, \\ & \quad - \{N_a \cdot M \cdot A \cdot B\}_{K_{as}} \rangle \end{aligned}$$

Thus, $tr_A \not\approx_E tr_I$. Specifically, A can observe the following difference

$$\begin{cases} \{N_a \cdot M \cdot A \cdot B\}_{K_{as}}\sigma_0 \neq_{E_{dy}} x\sigma_0 \\ \{N_a \cdot M \cdot A \cdot B\}_{K_{as}}\sigma_1 =_{E_{dy}} x\sigma_1 \end{cases}$$

where $\sigma_0 = [\{N_a \cdot K_{ab}\}_{K_{as}}/x]$ and $\sigma_1 = [\{N_a \cdot M \cdot A \cdot B\}_{K_{as}}/x]$. This difference suggests that we can simply add a new *check* event immediately after the *receive* event to prevent the attack. Thus, the new implementation strand of role A becomes

$$\begin{array}{l} \underline{A[M, A, B, S, N_a, K_{as}, x]} \\ \langle \{M, A, B, S, N_a, K_{as}\}, \\ \quad + M \cdot A \cdot B \cdot \{N_a \cdot M \cdot A \cdot B\}_{K_{as}}, \\ \quad - x_4,\ check(\{N_a \cdot M \cdot A \cdot B\}_{K_{as}} \neq x)\rangle \end{array}$$

The core innovation of our refinement is to add inequality check events to disallow such execution traces in TR_I that are not in compliance with protocol narration TR_0. Nonetheless, not all attack scenarios are useful to refine a protocol implementation, especially if the execution traces of the attack are in compliance with the protocol narration. For instance, the well-known man-in-the-middle attack due to Lowe [25] on the Needham-Schroeder public-key authentication protocol [31] can not be thwarted by adding any *check* event(s).

In general, a known attack can be categorized into the following three types:

- *type-I* attack, if all execution traces are in compliance with the ideal implementation. From a protocol implementor's point of view, this type of attack cannot be detected/prevented unless the design of the protocol is changed;
- *type-II* attack, if all execution traces are in compliance with the prudent implementation, and there exists an execution trace that is not in compliance with the ideal implementation;
- *type-III* attack, if there exists an execution trace that is in compliance with the coarse implementation, but not in compliance with the prudent implementation;

To the end of this section, we draw a picture of the classification of protocol implementations and attacks, as shown in Figure 3.

6. DISCUSSION AND RELATED WORK

Starting with the early work of Carlsen [11], a lot of efforts have been made to formalize security protocol descriptions or to devise semantics for them [10, 9, 13]. As pointed out by Abadi [1], how principals check incoming messages is an essential part of protocols, which is often neglected in protocol narrations.

Accordingly, many approaches from this line of research have striven to make such checks explicit. The treatments, however, are often either ad hoc and/or made in a case-by-case fashion, specialized for the Dolev-Yao style primitives.

Carlsen [11] defines four primitive security-relevant internal actions that can be generated from protocol narrations in a straightforward way. Even so, the actions checkvalue, which require accompanying type information to each word, are not always feasible. Caleiroa et al. [10] enumerate rules to characterize a principal's view of a message. Checks can be done on a message that is viewed as "reachable". The whole procedure is rather complex, which involves further concepts such as *analyzable position* and *inner facial pattern face*. Briais and Nestmann [9] identify three types of checks, which can be reduced to normal equality tests. The core technical innovation is to saturate a knowledge set first using Analysis rules and then compare it with the knowledge set obtained by Synthesis rules. The procedure coincides with the one given in [24] to decide recognizability under Dolev-Yao model. However, since the Analysis and Synthesis rules are specialized for Dolev-Yao model, it is not clear how to generalize the results to support algebraic properties in protocol narrations [30]. In [28, 8] checks are discussed informally and thus they do not automate this process. Besides, same as in [11] only structured data rather than bit strings are considered, which raises implementation issues in practice.

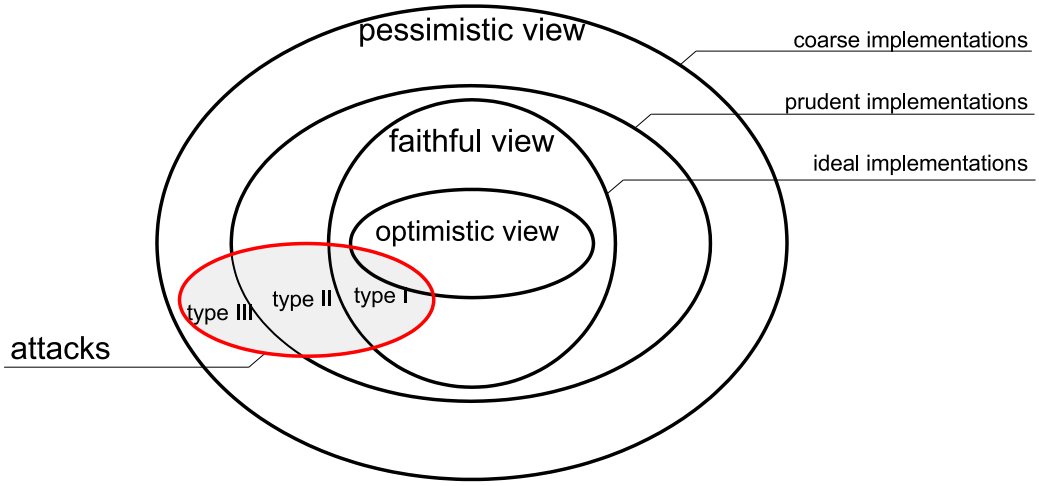
A major drawback of these approaches has been the lack of an intuitive, yet general, justification for such checks in a protocol narration. Thus, it is far from clear that all necessary checks are properly found in these approaches. Even though it is claimed in [9] that the maximum checks are derived from protocol narrations, there is no consensus on what are the maximum checks.

The main reason for the lack of intuitive justifications is that, compared to one's ability to interpret a message, a principal's *inability* to interpret a message is not well understood. In [17, 28, 5, 19], messages that cannot be interpreted with the principal's knowledge are treated as "black-boxes". This simplification may fail to give a precise semantics to a protocol, because relationship between those messages, such as $\mathtt{hash}(N_b)$ and N_b in the ASW protocol, could be missed. In [10], the notion of *transparent* and *opaque* messages resemble our notions of recognizable and unrecognizable terms, respectively. However, the definition of these notions is sound but not complete in a sense that a transparent message is recognizable but not vice versa. As an example, suppose that Alice knows $\{\{N_b\}_{K_{bs}}\}$ and she receive a message that is intended to be $N_b \cdot K_{bs}$. Then, $N_b \cdot K_{bs}$ is recognizable, that is, $\langle \{\{N_b\}_{K_{bs}}, x\}, [N_b \cdot K_{bs}/x]\rangle \rhd_{E_{dy}} N_b \cdot K_{bs}$. This is because $\mathtt{senc}(\mathtt{fst}(x), \mathtt{snd}(x))\sigma =_{E_{dy}} \{N_b\}_{K_{bs}}$ holds iff $x =_{E_{dy}} N_b \cdot K_{bs}$. This is usually referred as the "perfect encryption" assumption [4]. On the other hand, by the definition of $v_D(M)$ in [10], we have $v_{\{\{N_b\}_{K_{bs}}\}}(N_b \cdot K_{bs}) = v_{\{\{N_b\}_{K_{bs}}\}}(N_b); v_{\{\{N_b\}_{K_{bs}}\}}(K_{bs})$ and hence $N_b \cdot K_{bs}$ is not $\{\{N_b\}_{K_{bs}}\}$-transparent.

We build our work upon the concept of recognizability [24], which formalizes a principal's ability/inability to verify a message. Although it is initially proposed to understand type flaw attacks, the problem is similar to ours from a cognitive perspective. Nonetheless, for our purpose here, several extensions are required so as to provide a more fine-grained characterization of ambiguous terms.

It is fair to mention that the concept of *static equivalence* (on *frames*) in the applied pi calculus [3, 2] is similar in spirit to our operational equivalence (on markup term sets, Definition 3.2). But there is one essential difference: we discriminate unambiguous (ground term) and ambiguous (free variable) messages, whereas in static equivalence all messages are ambiguous. Naturally, the concept *observational equivalence* on processes corresponds to that of operational equivalence on strands.

Only recently, by Chevalier and Rusinowitch [13], has static equivalence been related to giving semantics to protocol narrations. To the best of our knowledge, this is the first result, with a convincing justification, that ensures all the possible checks are performed. However, since it only



Note: refined implementation = prudent implementations - type II attacks

	Attacks		
	Type-I	Type-II	Type-III
Ideal implementation	✓	×	×
Prudent implementation	✓	✓	×
Refined implementation	✓	×	×
Coarse implementation	✓	✓	✓

Figure 3: Classification of protocol implementations and attacks

allows equality checks, it does not support implementation refinement, as we do here.

7. CONCLUSION AND FUTURE WORK

In this paper, we provide a consensus view of security protocols for each group of people that amounts to the attacker's view. Specifically, we give ideal semantics to protocol narrations, by rigorously examining a principal's ability or inability to cope with potentially ambiguous incoming messages. The semantics are then used to guide protocol implementations in two complimentary ways. First, we derive a prudent implementation of a protocol, which performs all necessary equality checks and prevents type-III attacks. Second, we use type-II attacks to further refine a prudent implementation by performing additional new inequality checks. As such refinements are not feasible by either the protocol designers or the protocol verifiers alone, we motivate the interplay between protocol design and protocol verification via a semi-automated refinement process.

There are three major limitations of this study. First, although our results are not specialized for the Dolev-Yao intruder model, the accuracy of the semantics depends on how we model the principal's deduction capabilities. Failing to model the capabilities properly may result in unrealistic semantics. Second, the following questions arising in Section 3.3 are not answered:

(i). Under what conditions does there exist a constraint base of a markup term set $\vec{T}$?

(ii). How to determine and solve a constraint base if it exists?

Third, to simplify our discussion, we have treated fresh values (e.g., nonces and timestamps) as invariant data in one's initial knowledge. This is unrealistic in practice especially when a protocol execution involves multiple sessions.

Our future work will be aimed at addressing these limitations. In particular, we plan to investigate the problem of finding and solving constraint bases under more general equational theories. Besides, to overcome the inability of coping with fresh values, we will introduce a *new* event/node in extended strands; this would not affect our main results significantly.

8. REFERENCES

[1] M. Abadi. Security protocols and their properties. In *Foundations of Secure Computation, NATO Science Series*, pages 39–60. IOS Press, 2000.

[2] M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. *Theor. Comput. Sci.*, 367(1):2–32, 2006.

[3] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *POPL '01: Proceedings of the 28th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 104–115, New York, NY, USA, 2001. ACM.

[4] M. Abadi and M. R. Tuttle. A semantics for a logic of authentication (extended abstract). In *Proceedings of the tenth annual ACM symposium on Principles of distributed computing*, PODC '91, pages 201–216, New York, NY, USA, 1991. ACM.

[5] A. Armando, D. A. Basin, M. Bouallagui, Y. Chevalier, L. Compagna, S. Mödersheim, M. Rusinowitch, M. Turuani, L. Viganò, and L. Vigneron. The aviss security protocol analysis tool. In *CAV '02: Proceedings of the 14th International*

Conference on Computer Aided Verification, pages 349–353, London, UK, 2002. Springer-Verlag.

[6] N. Asokan, V. Shoup, and M. Waidner. Asynchronous protocols for optimistic fair exchange. In *Security and Privacy, 1998. Proceedings. 1998 IEEE Symposium on*, pages 86 –99, May 1998.

[7] B. Blanchet. Automatic verification of correspondences for security protocols. *J. Comput. Secur.*, 17(4):363–434, 2009.

[8] C. Bodei, M. Buchholtz, P. Degano, F. Nielson, and H. R. Nielson. Static validation of security protocols. *J. Comput. Secur.*, 13(3):347–390, 2005.

[9] S. Briais and U. Nestmann. A formal semantics for protocol narrations. *Theor. Comput. Sci.*, 389(3):484–511, 2007.

[10] C. Caleiro, L. Viganò, and D. Basin. On the semantics of alice&bob specifications of security protocols. *Theor. Comput. Sci.*, 367(1):88–122, 2006.

[11] U. Carlsen. Generating formal cryptographic protocol specifications. In *Proceedings of the 1994 IEEE Symposium on Security and Privacy*, SP '94, pages 137–146, Washington, DC, USA, 1994. IEEE Computer Society.

[12] P. Ceelen, S. Mauw, and S. Radomirović. Chosen-name attacks: An overlooked class of type-flaw attacks. *Electron. Notes Theor. Comput. Sci.*, 197:31–43, February 2008.

[13] Y. Chevalier and M. Rusinowitch. Compiling and securing cryptographic protocols. *Inf. Process. Lett.*, 110(3):116–122, 2010.

[14] J. Clark and J. Jacob. A survey of authentication protocol literature: Version 1.0, 1997.

[15] R. Corin and S. Etalle. An improved constraint-based system for the verification of security protocols. In *Proceedings of the 9th International Symposium on Static Analysis*, pages 326–341, London, UK, 2002. Springer-Verlag.

[16] C. J. Cremers. Unbounded verification, falsification, and characterization of security protocols by pattern refinement. In *CCS '08: Proceedings of the 15th ACM conference on Computer and communications security*, pages 119–128, New York, NY, USA, 2008. ACM.

[17] G. Denker and J. Millen. Capsl intermediate language. In *Proceedings of the Workshop on Formal Methods and Security Protocols — FMSP*, 1999.

[18] D. Dolev and A. Yao. On the security of public key protocols. *Information Theory, IEEE Transactions on*, 29(2):198–208, Mar 1983.

[19] A. Durante, R. Focardi, and R. Gorrieri. A compiler for analyzing cryptographic protocols using noninterference. *ACM Trans. Softw. Eng. Methodol.*, 9(4):488–528, 2000.

[20] F. Fabrega, J. Herzog, and J. Guttman. Strand spaces: why is a security protocol correct? In *Security and Privacy, 1998. Proceedings. 1998 IEEE Symposium on*, pages 160 –171, May 1998.

[21] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge*, volume 1 of *MIT Press Books*. The MIT Press, December 2003.

[22] J. D. Guttman and F. J. Thayer. Authentication tests and the structure of bundles. *Theor. Comput. Sci.*, 283:333–380, June 2002.

[23] Z. Li and W. Wang. Rethinking about type-flaw attacks. In *GLOBECOM 2010*, pages 1 –5, 2010.

[24] Z. Li and W. Wang. Deciding recognizability under dolev-yao intruder model. In M. Burmester, G. Tsudik, S. Magliveras, and I. Ilic, editors, *Information Security*, volume 6531 of *Lecture Notes in Computer Science*, pages 416–429. Springer Berlin / Heidelberg, 2011.

[25] G. Lowe. An attack on the needham-schroeder public-key authentication protocol. *Inf. Process. Lett.*, 56:131–133, November 1995.

[26] G. Lowe. Breaking and fixing the needham-schroeder public-key protocol using fdr. In *TACAs '96*, pages 147–166, 1996.

[27] G. Lowe. Some new attacks upon security protocols. In *Proceedings of the 9th IEEE workshop on Computer Security Foundations*, CSFW '96, pages 162–, Washington, DC, USA, 1996. IEEE Computer Society.

[28] G. Lowe. Casper: a compiler for the analysis of security protocols. *J. Comput. Secur.*, 6(1-2):53–84, 1998.

[29] J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *CCS '01: Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 166–175, New York, NY, USA, 2001. ACM.

[30] S. Modersheim. Algebraic properties in alice and bob notation. *Availability, Reliability and Security, International Conference on*, 0:433–440, 2009.

[31] R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21:993–999, December 1978.

[32] D. Otway and O. Rees. Efficient and timely mutual authentication. *SIGOPS Oper. Syst. Rev.*, 21(1):8–10, 1987.

[33] J. Robinson. *Handbook of Automated Reasoning (2 Volume Set)*. MIT Press, Cambridge, MA, USA, 2001.

[34] D. X. Song, S. Berezin, and A. Perrig. Athena: a novel approach to efficient automatic security protocol analysis. *J. Comput. Secur.*, 9(1-2):47–74, 2001.