REASONING ABOUT RECOGNIZABILITY IN SECURITY PROTOCOLS

by

Zhiwei Li

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Computing and Information Systems

Charlotte

2012

Approved by:

____
Dr. Weichao Wang

____
Dr. Xintao Wu

____
Dr. Mohamed Shehab

____
Dr. Aidong Lu

____
Dr. Mary Maureen Brown

ABSTRACT

ZHIWEI LI. Reasoning about recognizability in security protocols. (Under the direction of DR. WEICHAO WANG)

Although verifying a message has long been recognized as an important concept, which has been used explicitly or implicitly in security protocol analysis, there is no consensus on its exact meaning. Such a lack of formal treatment of the concept makes it extremely difficult to evaluate the vulnerability of security protocols.

This dissertation offers a precise answer to the question: *What is meant by saying that a message can be "verified"?* The core technical innovation is a third notion of knowledge in security protocols — recognizability. It can be considered as intermediate between deduction and static equivalence, two classical knowledge notions in security protocols. We believe that the notion of recognizability sheds important lights on the study of security protocols. More specifically, this thesis makes four contributions.

First, we develop a knowledge model to capture an agent's cognitive ability to understand messages. Thanks to a clear distinction between de re/dicto interpretations of a message, the knowledge model unifies both computational and symbolic views of cryptography gracefully.

Second, we propose a new notion of knowledge in security protocols — recognizability — to fully capture one's ability or inability to cope with potentially ambiguous messages. A terminating procedure is given to decide recognizability under the standard Dolev-Yao model.

Third, we establish a faithful view of the attacker based on recognizability. This yields new insights into protocol compilations and protocol implementations. Specifically, we identify two types of attacks that can be thawed through adjusting the protocol implementation; and show that an ideal implementation that corresponds to the intended protocol semantics does not always exist. Overall, the obtained attacker's view provides a path to more secure protocol designs and implementations.

Fourth, we use recognizability to provide a new perspective on type-flaw attacks. Unlike most previous approaches that have focused on heuristic schemes to detect or prevent type-flaw attacks, our approach exposes the enabling factors of such attacks. Similarly, we apply the notion of recognizability to analyze off-line guessing attacks. Without enumerating rules to determine whether a guess can be "verified", we derive a new definition based on recognizability to fully capture the attacker's guessing capabilities. This definition offers a general framework to reason about guessing attacks in a symbolic setting, independent of specific intruder models. We show how the framework can be used to analyze both passive and active guessing attacks.

ACKNOWLEDGMENTS

This dissertation would not have happened without the help, inspiration, and encouragement of many people along the way.

First and foremost, I want to thank my advisors: Prof. Weichao Wang and Prof. Aidong Lu. I am very fortunate to have the privilege to be their graduate student. Over the last several years, they have always been ready for guidance in my research and yet willing to step aside whenever I wished to explore on my own. Several of those late nights helping with my first research paper will remain forever unforgettable. Their support went beyond the role as my advisors and I couldn't have asked for more. When I had a newborn, they granted me the freedom to make my schedule flexible to fit my family's needs.

I am honored to have Professors Xintao Wu, Mohamed Shehab, and Mary Maureen Brown serve on my PhD Committee. They provided helpful comments to improve this manuscript. In particular, I would like to thank Prof. Xintao Wu for teaching me about privacy-preserving data mining (ITIS 8010), which brought a broader perspective to my research. More importantly, I wouldn't have come to UNC Charlotte if it wasn't Prof. Wu's research caught my initial attention[1]. I would also like to thank Prof. Mary Maureen Brown for offering me the opportunity to work with her in the summer 2010. Throughout that time, she always had more faith in my abilities than myself, provided invaluable advice, and kept challenging me to think and write clearly about problems encountered. She has taught me a great deal; I really appreciate it.

[1] I was interested in data mining before I came to the states.

I am grateful to Prof. Yuliang Zheng for the opportunity to discuss cryptography and doing research in general. Prof. Yuliang Zheng freely offered his time and advice, even though I was not his student.

I benefited greatly from wonderful contacts outside of UNC Charlotte as well. I am in debt to Véronique Cortier, my terrific mentor at CNRS/Loria-INRIA laboratory for comments, criticism, and helpful discussions on my work, and for funding me to attend the Computer Security Foundations Symposium. Thanks to Ben Smyth for helping me with the applied pi calculus, and Vincent Cheval for wonderful conversations about observational equivalence at late nights.

A very special thanks to Prof. Dawn Song for her papers on Athena ignited my interest in formal security protocol analysis. I used to be interested in data mining before I came to UNC Charlotte, and my advisor gave me her papers to read when I started my PhD study here. I really got poisoned by Athena and decided to explore my interests in security protocol analysis thereafter.

Last but not least, I would like to thank my family. My parents' encouragement and supports are the backbones of my constant struggle from frustration to empowerment. I am thankful to my in-laws for being incredibly kind, generous, and supportive to their son in law. Most of all, thanks to my wife, Li Wu, who brought a home to me, and gave birth to our beautiful son, Castor. Thanks to Castor for being a best excuse for me to take breaks. Finally, to honor her memory, I dedicate this thesis to my grandmother, Guizhu Huang, whose moral values and beliefs imparted to me throughout my life.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

With the ever-increasing diversity of networked and distributed systems, protocols are widely deployed to make communication between different computing systems possible. And yet, security protocols are used to ensure these communications are not abused by providing secure services, including authentication, confidentiality, secrecy, and privacy. Unfortunately, security protocols are notoriously error-prone and some attacks may take years or even decades to discover [95, 81]. This is because, on one hand, security protocols are intricate and an expected protocol execution naturally leads designers to ignore other possible protocol executions; and on the other hand, the attacker is powerful to intercept, eavesdrop, and modify communication between network entities.

Over the last 30 years, formal methods [90, 86, 88] have played an important role in finding attacks on security protocols. In formal security protocol analysis, we noticed that the term "verify" has been used either explicitly or implicitly under different scenarios. For example, an off-line guessing attack is feasible only if a correct guess can be *verified*. Let us consider the following simple one-way authentication protocol:

$$\text{Message 1.} \quad A \rightarrow B : \{N_A\}_{K_{AB}}$$

$$\text{Message 2.} \quad B \rightarrow A : \{\mathtt{f}(N_A)\}_{K_{AB}}$$

The protocol tells the story where principal A wants to authenticate itself to principal B. Here N_A is a fresh random number (i.e., nonce) generated by A and K_{AB} is the symmetric key shared between A and B, and $\mathtt{f}$ is a given and known function (e.g., $\mathtt{f}(N_A) = N_A + 1$). An attacker may verify a guess of K_{AB}, say g, by decrypting both messages with the guessed key g. Suppose that the decryption results are r_1 and r_2, respectively. Then, g is the correct guess, if r_2 equals $\mathtt{f}(r_1)$.

Similarly, an attacker can launch a type-flaw attack only if some protocol principal is unable to *verify* incoming message(s). To further elaborate this point, let us consider the concrete example of the Otway-Rees protocol [96]:

Message 1. $A \to B : M, A, B, \{N_A, M, A, B\}_{K_{AS}}$

Message 2. $B \to S : M, A, B, \{N_A, M, A, B\}_{K_{AS}}, \{N_B, M, A, B\}_{K_{BS}}$

Message 3. $S \to B : M, \{N_A, K_{AB}\}_{K_{AS}}, \{N_B, K_{AB}\}_{K_{BS}}$

Message 4. $B \to A : M, \{N_A, K_{AB}\}_{K_{AS}}$

In this protocol, two principals A and B are both connected to a trusted third party S with whom they share the symmetric keys K_{AS} and K_{BS}, respectively. After executing the first three messages, principal A is expecting a symmetric key K_{AB} shared between A and B, from the trusted third party S. As the shared key K_{AB} is dynamically generated by S, A does not have any prior knowledge of the bit string. In other words, A is unable to *verify* a message of the form $M, \{N_A, t\}_{K_{AS}}$, as long as the bit string length of t equals to that of K_{AB}. Therefore, an attacker can easily replay the message $\{N_A, M, A, B\}_{K_{AS}}$ to A and thus A would use M, A, B as the shared symmetric key between A and B, as long as the length satisfies the requirement.

However, most of the previous work use the term "verify" in an ad-hoc manner and the term means differently in different contexts.

In efforts to find guessing attacks, "verify" is a term widely accepted to characterize a correct guess and thus many approaches focus on heuristics to explore ways of verifying a guess [35, 85, 59]. This is usually done by enumerating rules to determine whether a guess can be "verified". These rules are used to derive an inference system modeling the guessing capabilities [44], by extending the standard Dolev-Yao model [48]. Realizing the "incompleteness" of such an inference system in a sense that it may fail to capture some "verifiable" guess, Drielsma et al. [49] develop a precise formalization of off-line guessing attacks, which is independent of any particular intruder model. However, no automatic procedure is given in [49] and, more importantly, it only allows guessing/verifying atomic values.

To defend against type-flaw attacks, Catherine Meadows [89] develops a formal model of types to characterize one's capability to verify messages. Without exploring the intuitive idea behind, the procedure of verifying the locality of types could be rather complicated. More importantly, it fails to capture a principal's inability to verify a message precisely. In [79, 78], Z specification language is employed to model ambiguous messages. The approach based on Z specification language cannot be directly applied to existing protocol analysis tools in a straight-forward way.

In security protocol compilation, messages that cannot be verified are treated as "black-boxes" [45, 84, 7, 50]. This simplification ma fail to give precise semantics to protocol narrations. Caleiroa et al. [21] enumerate rules to characterize a principal's view of a message. A message can be "verified" is viewed as "reachable". The whole procedure is rather complex, which involves further concepts such as *analyzable position* and *inner facial pattern face*. The notions of *transparent* and *opaque* messages are further proposed to characterize "verifiable" and "unverifiable" messages, as we do here. However, the method used to define such notions is heuristic rather than a conceptual basis and hence the definition of these notions is sound but not complete in a sense that a transparent message is "verifiable" but not vice versa.

Despite considerable efforts to understand how to verify a message, there is no consensus on the definition of the term "verify" in the first place. Such a lack of generic definition makes it extremely difficult to evaluate the vulnerability of security protocols. In this thesis, we therefore pursue a satisfying answer to the very first question: *What is meant by saying that a message can be "verified"?*

1.1 Contributions

In this work, we provide a precise answer to the question: "what is meant by saying that a message can be verified" by developing a new knowledge notion – recognizability — in security protocol analysis. More specifically, we make the following contributions:

- We define a new knowledge notion – recognizability – to characterize a principal's ability/inability to cope with ambiguous messages. Informally, we say a principal is able to *recognize* a message, if he has certain expectation about its bit string representation [57]. That is, given a bit string t, though he may not necessarily know t, he can verify that whether or not it is the bit string representing the expected message.
- We give a procedure to decide recognizability under the widely used Dolev-Yao intruder model. We intend to extend such results to more general equational theories.

- We apply the notion of recognizability to analyze type-flaw attacks. This enabled us to provide a consensus view of security protocols by eliciting both operational and denotational semantics of protocols. More importantly, we show that the security of a protocol can be enhanced by engaging both protocol designers and verifiers via a semi-automatic semantic refinement process.
- Based on the notion of recognizability, we propose a new definition to fully and faithfully capture the attacker's guessing capabilities. This provides a general framework to reason about guessing attacks in a symbolic setting, independent of specific intruder models. We show how the framework can be used to analyze both passive and active guessing attacks.

1.2 Related Works

The new notion of recognizability is closely related to the classical notions of knowledge in security protocols: deducibility [82] and indistinguishability [3].

Deducibility is one kind of algorithmic knowledge [63], in which "knowing what" can be determined by an algorithm. Due to its simplicity, Halpern and Pucella have successfully used algorithmic knowledge to model several different adversaries [64]. Then Pucella proves that the decision problem in a general case is NP-complete [98]. Our work is also inspired by their previous research on algorithmic knowledge.

The BAN [20] logic, proposed by Burrows, Abadi and Needham, is based on the deducibility notion of knowledge. It is probably the first extensively studied logic in protocol analysis based on knowledge. The agent's capability to synthesize messages is directly modeled by a set of inference rules, which are used to determine implicit knowledge. It is difficult to apply BAN logic to dynamically evolving knowledge to establish a general model. There are many other logics introduced in security protocol analysis [57, 77, 104]. We find that most approaches using Dolev-Yao style adversary are based on the deducibility notion.

The concept of indistinguishability comes directly from the classical possible-worlds approach to model knowledge [54], in which the actual world is considered to be one of many possible worlds. In security protocol analysis, message $\{N_A\}_{K_{AB}}$ and message $\{N_B\}_{K_{AB}}$ are indistinguishable if one does not know K_{AB} and has not seen those messages before. Recently, Cohen and Dam [29] provide a generalized Kripke semantics for studying this type of knowledge in security protocol analysis. They use static equivalence [4] to capture the indistinguishability for agents. Abadi and Cortier [3] examine the decidability of these two notions of knowledge by studying the underlying equational theories for deduction and static equivalence. This is especially important since the termination of analysis of the knowledge might not be guaranteed when decidability result is not held. Our approach circumvents this decidability problem since "knowing what" can always be determined by an algorithm. Following this line of research, new decidability results are obtained for monoidal equational theories [36].

Though closely related to these two classical notions of knowledge, our notion of recognizability is fundamentally different from deducibility and indistinguishability. For example, we assume that Alice knows $\{N_B\}_{K_B^+}$ and Bob's public key K_B^+. Then even Alice does not know the message N_B, she can still verify whether or not a given message is in fact N_B by simply encrypting it with the public key of Bob (i.e., K_B^+)

and comparing the result with $\{N_B\}_{K_B^+}$ which is "deducible" from her knowledge.

For static equivalence, procedures are given to decide whether two given messages are statically equivalent [3, 37, 27]. Regarding our notion of recognizability, we concern with the problem: given a message m whether there exists another message m' that is indistinguishable from m (by the observer). Since the other message is not provided beforehand, deciding recognizability and static equivalence could be significantly different. As we have seen in the previously mentioned type-flaw attack to the Otway-Rees protocol, the last message $M, \{N_A, K_{AB}\}_{K_{AS}}$ is forged by an attacker with the message $M, \{N_A, M, A, B\}_{K_{AS}}$. Given those two messages, it can be easily shown that they are statically equivalent in the applied pi calculus and thus $M, \{N_A, K_{AB}\}_{K_{AS}}$, or more precisely $\{N_A, K_{AB}\}_{K_{AS}}$, is not "verifiable" by the protocol participant. However, without this hindsight, it is not straightforward to see whether or not $M, \{N_A, K_{AB}\}_{K_{AS}}$ is "verifiable".

1.3 Outline

In Chapter 2 and 3, we introduce a new notion of knowledge in security protocols — recognizability. In Chapters 4 and 5, we propose a constraint based approach to decide recognizability under the widely used Dolev-Yao intruder model. In Chapters 6 and 7, we apply the notion of recognizability to security protocol compilation and the analysis of type-flaw attacks and off-line guessing attacks. Chapter 8 concludes this thesis.

CHAPTER 2: A NEW KNOWLEDGE MODEL

Before diving into the question upfront, we should first formalize the exact meaning of "knowing a message". In the literature, there are types of formalisms corresponding to both computational and symbolic views of cryptography. In a computational view [9, 5] it means one possesses some piece of bit string, whereas in a symbolic view a message is understood as a term structure [48, 93, 38].

The lack of a unified view prohibits a faithful account of knowledge in security protocol analysis. To see this, let us consider a well-known argument of Abadi and Cortier [3]:

> *Suppose that we are interested in a protocol that transmits an encrypted Boolean value v, possibly a different one in each run. We might like to express that this Boolean value remains secret by saying that no attacker can learn it by eavesdropping on the protocol. On the other hand, it is unreasonable to say that an attacker cannot deduce the well-known Boolean values true and false.*

Here, discrepancy arises due to the unclear meaning of "knowing the Boolean value v". Indeed, the Boolean value of v (either true or false) is known, and thus v is known. This, however, contradicts with common sense reasoning, because one is still unable to *determine* whether v *is* true or false.

At this point, one might be lead to believe that "knowing message m" means "being able to determine the value of message m". Unfortunately, this interpretation may still not comply with common sense.

According to the Merriam-Webster dictionary [1], the term "determine" is defined as "to *fix* the form, position, or character of beforehand". In our context, the form is simply a message, and "to fix the form" does not necessary mean knowing the message. To further elaborate this point, let us consider the following simplified login protocol, which we use everyday to authenticate ourselves to websites:

$$\text{Message 1.} \quad C \to S : C, \texttt{hash}(P_C)$$

$$\text{Message 2.} \quad S \to C : result$$

Here, $\texttt{hash}()$ is assumed to be a collision-free and one way function. Whenever the client C wants to login to the web server S, it sends a message with both its username C and hashed password $\texttt{hash}(P_C)$ to S. On the server side, S maintains a list of usernames $\{C_1, C_2, \cdots, C_n\}$ and their corresponding credentials (i.e., hashed passwords) $\{H_1, H_2, \cdots, H_n\}$. What happens is that, the server S verifies whether $C = C_i$ and $\texttt{hash}(P_C) = H_i$ for some i after receiving the first message. For a legitimate login attempt, the server S does find an i such that $C = C_i$ and $\texttt{hash}(P_C) = H_i$. Therefore, the server is able to determine the value of C and P_C in a sense that the form is fixed. Indeed, since the server has access to C_i, it also knows C. However, for P_C, it is unreasonable to say that the server S knows P_C, despite the fact that its value is determined. Actually, determining the value (or equivalently correctness) of P_C without disclosing its value is a basic design guideline for implementing password authentication.

The above discrepancy between intuition and formalism is an instance of a *de re/*

de dicto ambiguity [105, 68, 29]. Let's look at the following sentence:

"Alice knows the value of message m."

Under the *de re* reading, it means that there exists a value x such that x is the value of message m and Alice knows x, that is,

$$(\exists x)(m \text{ has the value of } x \wedge \text{Alice knows } x) \tag{i}$$

Under the *de dicto* reading it means that there exists a value x such that x is the value of message m and Alice knows the fact that x is the value of m, that is,

$$(\exists x)(m \text{ has the value of } x \wedge \text{Alice knows the fact that } m \text{ has the value of } x) \tag{ii}$$

In other words, Alice is able to determine (but not necessarily to know) the value of m. Note that (ii) is different from the following trivial condition:

$$\text{Alice knows } (\exists x)(m \text{ has the value of } x)$$

In this chapter, we show how the *de re/ de dicto* dichotomy gives rise to a novel knowledge model of agents. Note that we use "agent" to mean a legitimate protocol participant, the attacker, or simply a principal (— we use the these terms interchangeably in this thesis). Unlike most existing epistemic approaches in security protocol analysis [20, 99, 41] that aim to verify security protocols, our primarily goal is to capture an agent's cognitive ability to understand messages. The reason is two-fold.

First, as a security protocol is essentially a message-passing system [54] with two primitive actions *send* and *receive*, agent's knowledge should be fully characterized by the messages he possessed and received. Hence, understanding those messages is a crucial component of security protocol analysis.

Second, this primitive goal frees us from the need to model security protocols, which is usually done by transition systems [91, 58, 41, 66] and is rather involved.

We can thus restrict our attention to single agents and leave their interaction with environment implicit, rather than to consider a multi-agent system in its most general form. We remark that, although modeling intruder's capabilities [32, 64, 38, 70] is of interest on its own, in this thesis we will not distinguish between legitimate protocol participants and the attacker.

The remainder of this chapter is organized as follows. We start with the *de re* interpretation of a message. Then, we turn to the *de dicto* interpretation of a message. Next, we build a new knowledge model on top of the *de re/ dicto* interpretation. Before concluding this chapter, we show through several examples how the knowledge model can be used in formal security protocol analysis.

2.1 The *de re* Interpretation

We have used the term "know" rather informally without a precise definition. As discussed before, a precise meaning of "knowing a message" involves both the *de re* and *de dicto* interpretations of the message. In this section, we formalize the *de re* interpretation of a message, that is knowing the bit string value, and deter the *de dicto* interpretation to the next section. As we will see, such knowledge of an agent is a form of algorithmic knowledge [54], and can be modeled by deducibility relation.

2.1.1 Symbolizing Bit Strings

Although an exchanged message is simply a bit string in real protocol execution, in protocol specification it is often represented as expression defined in some term algebra. After a brief review term algebra, we show how bit strings can be manipulated symbolically without losing accuracy. We mainly follow the notation in [46].

2.1.1.1 Term Algebra

A *signature* is a finite set of function symbols $\mathcal{F}$ and a possibly infinite set of constants $\mathcal{A}$. We discriminate *public* and *private* function symbols, respectively denoted by $\mathcal{F}^+$ and $\mathcal{F}^-$. Public functions are used to describe operations that can be freely performed by a principal, and private functions are used to constrain the relation between terms. Each function symbol has an associated arity.

Let $\mathcal{X}$ be a possibly infinite set of variables. Then, *term algebra* $\mathcal{T}(\mathcal{F}, \mathcal{A}, \mathcal{X})$ is defined 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 $t_1 =_s t_2$.

We say that s is a *subterm* of t, written $s \subseteq t$, if either $s =_s t$ or $t =_s f(t_1, \cdots, t_n)$ and s is a subterm of t_i for some i. We also write $s \subset t$ to mean $s \subseteq t$ and $s \neq_s t$. A term s *occurs* in a term set T if $s \subseteq u$ for some $u \in T$. The *size* of a term t is defined as

$$\|t\| \triangleq \begin{cases} 1 & \text{if } t \in \mathcal{X} \cup \mathcal{A} \\ 1 + \sum_{i=1}^{n} \|t_i\| & \text{if } t = f(t_1, \cdots, t_n) \end{cases}$$

For term set T, we define $\|T\|$ as $\sum_{t \in T} \|t\|$. We define inductively the *immediate subterm set* of a term t, denoted by $sub(t)$, as follows:

- If $t =_s f(t_1, t_2 \cdots, t_n)$ and $n > 0$, then $sub(t) = \{t_1, t_2, \cdots, t_n\}$;
- otherwise, $sub(t) = \{t\}$.

For convenience, we use $ff(t)$ to indicate the outmost function symbol of t and let $ff(t) = \phi$ if $\|t\| = 1$.

We will use l, r, s, t to denote terms and x, y, z to denote variables. As usual, $fn(t)$ and $fv(t)$ are defined as the set of constants and variables that occur in term t respectively. A term is said to be *ground* when $fv(t) = \emptyset$. Theses notations are extended as expected to sets of terms. We tend to use the words "term" and "message" interchangeably in the rest of this thesis.

A *context* C is a term with exactly one "hole" $\square$. Then the term $C[t]$ is C except $\square$ is replaced by t. A *substitution* is a finite tuple $[t_1/x_1, ..., t_n/x_n]$ mapping from variables x_i to terms t_i, and will generally be represented by σ, θ, μ, or η. 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 use ϵ to denote an *empty substitution*, that is $Dom(\epsilon) = \emptyset$. A substitution σ is *ground* if $Ran(\sigma)$ is a ground term set. We write $\sigma = \theta$ (resp. $\sigma =_E \theta$) if $Dom(\sigma) = Dom(\theta)$ and $x\sigma =_s x\theta$ (resp. $x\sigma =_E x\theta$) for all $x \in Dom(\sigma)$. We define the *composition* of substitutions σ and θ as a new substitution $\sigma \circ \theta$ (or simply $\sigma\theta$) such that $t\sigma \circ \theta =_s (t\sigma)\theta$. We say that σ is *more general* than θ, notation $\sigma \preceq \theta$, if $\theta = \sigma\eta$ for some substitution η. We write $mgu(s, t)$ for the most general unifier of s and t.

The following function symbols are widely used in formal security protocol analysis.

$$\mathcal{F}^+_{dy} = \{\texttt{pair}, \texttt{senc}, \texttt{penc}, \texttt{hash}, \texttt{fst}, \texttt{snd}, \texttt{sdec}, \texttt{pdec}\}$$

$$\mathcal{F}^-_{dy} = \{\texttt{pk}, \texttt{sk}\}$$

There are

- four public constructive function symbols for encryption (i.e., `senc` and `penc` for symmetric and asymmetric encryption, respectively), concatenation (i.e., `pair`), and hashing (i.e., `hash`);
- four public destructive function symbols for decryption (i.e., `sdec` and `pdec` for symmetric and asymmetric encryption, respectively) and split (i.e., `fst` and `snd`);
- two private function symbols `pk` and `sk` to denote a public key and a private key, respectively.

To reduce notational clutter, we will often 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 denotes $(((t_1 \cdot t_2) \cdot t_3) \cdot \cdots \cdot t_n)$. Additionally, $\{s\}_t$ denotes $\mathtt{penc}(s,t)$ if t is either a public key or a private key, and $\mathtt{senc}(s,t)$ otherwise.

2.1.1.2 Equational Theory

Note that a bit string may correspond to several syntactically different terms. For example, the bit string value of $a \oplus b$ is the same as that of $b \oplus a$, where $\oplus$ denotes exclusive or. In formal security protocol analysis, we use an equational theory to capture such "equalities". More precisely, 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. For convenience, we let $E^{\mathcal{L}} = \{r | l = r \in E\}$.

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 $x \in X$.

As an example, we encode the standard Dolev-Yao model [48] by the following equational theory E_{dy}.

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

Figure 1: Equational Theory E_{dy} modeling the standard Dolev-Yao intruder.

Although we can abstract away bit string value of a term in protocol specification, the bit string may be relevant for further protocol analysis. For example, regarding the Abadi and Cortier's argument, we can use $\{v\}_{K^+}$ to represent the encrypted Boolean value v. If v is treated merely as a symbol, we will not know whether it is a Boolean value or a 128-bit value, and thus it is unreasonable to say v known. Now, it is not hard to see that a technical reason for the discrepancy is that v is treated both as a bit string (of length 1) and a symbol v, whereas the knowledge reasoning failed to capture this. Consequently, it is highly desirable to handle both symbolic expression and bit string in a uniform way.

We thus introduce a special set of constant symbols $\mathcal{N} = \{n_i | i = 0, 1, 2, \cdots\}$ such that the bit string value for each $n_i \in \mathcal{N}$ is i, yielding a new term algebra $\mathcal{T}(\mathcal{F}, \mathcal{A} \cup \mathcal{N}, \mathcal{X})$. Then, we can use equations to assign values to terms. For example, in order to describe the fact v is true, we can add $v = n_1$ into the underlying equation theory.

2.1.1.3 Rewriting Systems

Let $\rightarrow$ be a binary relation. As is commonplace, the transitive closure and reflexive transitive closure of $\rightarrow$ are denoted by $\rightarrow^+$ and $\rightarrow^*$ respectively. We say that an element p is *reducible* for $\rightarrow$ if there is an element q such that $p \rightarrow q$ and *irreducible* otherwise. If $p \rightarrow^* q$, and q is irreducible for $\rightarrow$, then q is called a $\rightarrow$*-normal form* of p. We write $p \rightarrow^! q$ if $p \rightarrow^* q$ and q is a $\rightarrow$-normal form.

We say that $\rightarrow$ is *terminating* or *well-founded* if there exists no infinite derivation $q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow \cdots$. $\rightarrow$ is *confluent* if there is an element q such that $q_1 \rightarrow^* q$ and $q_2 \rightarrow^* q$ whenever $q_0 \rightarrow^* q_1$ and $q_0 \rightarrow^* q_2$.

A *term rewriting system* R consists of a set of rules, $l \rightarrow r$, where both l and r are terms. A term rewriting system R defines a *term rewriting relation* $\rightarrow_R$ in a standard way as follows: $C[l\sigma] \rightarrow_R C[r\sigma]$ where C is a context, $l \rightarrow r \in R$, and σ is a substitution such that $Dom(\sigma) \subseteq fv(l)$. It should be pointed out that we often require $fv(l) \cap fv(C) = \emptyset$ and thus $C[l]\sigma \rightarrow_R C[r]\sigma$. If $fv(l) \cap fv(C) \neq \emptyset$, then we could use variable renaming to resolve this conflict. For a given term rewrite relation $\rightarrow_R$, we also write R-normal instead of $\rightarrow_R$-normal. Given an equational theory E, we define R_E by $R_E = \{l \rightarrow r | l = r \in E\}$. When $\rightarrow_{R_E}$ is confluent, $t_1 =_E t_2$ if and only if t_1 and t_2 have the same R_E-normal form.

Theorem 2.1.1 (Birkhoff's Theorem [11]). $s =_E t$ if and only if $s \leftrightarrow^*_{R_E} t$.

2.1.2 Explicit and Implicit Knowledge

Before proceeding, let us re-examine the Abadi and Cortier's argument. The rationale is as follows: v is a Boolean value and Boolean values (either true or false) are well-known, so v is deducible. Now, we let v be a 2-bit value. Then, because v is a 2-bit value and 2-bit values $(0, 1, 2, 3)$ are well-known, we conclude that v is also deducible. Likewise, a 128-bit value v should also be deducible. We reach an obvious contradiction to our intuition.

The astute reader may argue that this is because the 128-bit values are not *well-known*. The question is: What makes 1-bit values well-known and 128-bit values not well-known, considering the fact that 128-bit values can easily be obtained by enumeration? Or simply, what is the difference between *known* and *well-known*?

Intuitively, one must be *aware* of a value before it becomes *well-known*. Convincingly, one tends to be much less aware of 128-bit values than Boolean values. To account for the notion of awareness [54], we use a ground term set T to represent what one explicitly knows and is aware of. We refer to this type of knowledge as *explicit knowledge*, and use *implicit knowledge* to mean knowledge computed from one's explicit knowledge.

The most straightforward way to model the attacker's implicit knowledge is in terms of message deducibility [48, 82]. That is, given a term set representing one's explicit knowledge, one can compute a term t from T. More precisely, the deduction relations $\vdash$ and $\vdash_E$ are defined as follows.

$$\boxed{\vdash^{(n)}} \qquad \text{(R1)} \quad \frac{t \in T}{T \vdash^{(1)} t}$$

$$\text{(R2)} \quad \frac{T \vdash^{(n_1)} t_1 \cdots T \vdash^{(n_k)} t_k}{T \vdash^{(1+\max\limits_{1 \leq i \leq k} n_i)} f(t_1, \cdots, t_k)} \; f \in \mathcal{F}^+$$

$$\boxed{\vdash_E^{(n)}} \qquad \text{(R3)} \quad \frac{T \vdash^{(n)} t}{T \vdash_E^{(n)} t}$$

$$\text{(R4)} \quad \frac{T \vdash^{(n)} s \quad s =_E t}{T \vdash_E^{(n+1)} t} s \neq_s t$$

We say that t can be deduced from T, written $T \vdash t$, if $T \vdash^{(n)} t$ for some n. Similarly, we say that t can be deduced from T under equational theory E, written $T \vdash_E t$, if $T \vdash_E^{(n)} t$ for some n. Two term sets S and T are *equivalent* (under E), denoted as $S \equiv_E T$, if $S \vdash_E t$ for every $t \in T$ and $T \vdash_E s$ for every $s \in S$.

In general, $\vdash_E$ can be undecidable. Moreover, Abadi and Cortier [3] showed that even when equality is decidable $\vdash_E$ can still be undecidable. Note that the only difference between $\vdash$ and $\vdash_E$ is that the latter considers an equational theory E, whereas $\vdash$ does not. So, the computational cost of deciding $\vdash$ is considerably lower than that of $\vdash_E$.

Proposition 2.1.2. Both relations $\vdash$ and $\vdash_E$ are closed under substitution.

Due to rule (R1) one's explicit knowledge is also part of its implicit knowledge. Moreover, the definition of $\vdash_E$ usually permits an algorithm [38] to determine messages that one explicitly or implicitly knows. For this reason, the knowledge under the *de re* interpretation can be seen as a type of algorithmic knowledge [64].

Thanks to the following lemma, computation involved in establishing $\vdash_E$ can be characterized by some recipe.

Lemma 2.1.3 (Recipe Lemma). Let T be a term set and σ be a substitution. Then,

$T\sigma \vdash_E t$ if and only if $T \vdash u$ for some u, called *recipe*, such that $u\sigma =_E t$.

Proof. The "if" part of the lemma is obvious, because $\vdash_E$ is closed under substitution by Proposition 2.1.2. We now prove the "only if" part. By the definition of $\vdash_E$, we have $T\sigma \vdash_E t$ if and only if $T\sigma \vdash s$ for some s such that $s =_E t$.

Suppose that $T\sigma \vdash^{(n)} s$. We proceed by induction on n. For the base case, $n = 1$, by the definition of $\vdash$ we thus have $s \in T\sigma$. Then, there is a term $u \in T$ such thas $u\sigma =_s s =_E t$. The claim is true. Now, we suppose that $T\sigma \vdash^{(n)} s$ implies $T \vdash u$ for some u such that $u\sigma =_s s$ whenever $n \leq k$.

For $n = k + 1$, using the definition of $\vdash$ we observe that $T\sigma \vdash sub(s)$ and $ff(s) \in \mathcal{F}^+$. Let $s =_s f(s_1, \cdots, s_m)$ and $T\sigma \vdash^{(n_i)} s_i$. Since $n_i \leq k$, by induction hypothesis, we know that for each $s_i \in sub(s)$ there exists a term s'_i such that $T \vdash s'_i$ and $s'_i\sigma =_s s_i$. By letting $u =_s f(s'_1, \cdots, s'_m)$, we see that $T \vdash f(s'_1, \cdots, s'_m)$ and thus$f(s'_1, \cdots, s'_m)\sigma =_s s =_E t$. This completes the proof. □

Proposition 2.1.4 (Perfect Encryption). Let s and t be two terms that occur in term set T.

(i). Suppose that the only occurrence of s in T is $\{s\}_{K^+}$. Then, $T \vdash_{E_{dy}} s$ if and only if $T \vdash_{E_{dy}} K^-$.

(ii). Suppose that the only occurrence of s in T is $\{s\}_K$. Then, $T \vdash_{E_{dy}} s$ if and only if $T \vdash_{E_{dy}} K$.

The above proposition asserts that no one can learn a secret s from its encryption without the decryption key (either symmetric key K or asymmetric key K^-); this is the so-called *perfect cryptography assumption*, which is widely used in formal security protocol analysis.

Definition 2.1.5 (Ground *de re* Knowledge). Let E be an equational theory and T be a ground term set. We say a ground term t is *de re* known in model (T, E), written $(T, E) \models \mathbf{K}\texttt{re}(t)$, if and only if $T \vdash_E t$.

We stress that under the *de re* reading a message is merely a bit string. Knowing a message only means that one possesses or is able to compute its bit string representation; the agent has no information about the meaning of the bit string, which is the subject of the next section.

Example 1. Let $T = \{\{N_B\}_{K_B^+}, K_B^-\}$ represent Alice's explicit (*de re*) knowledge. Since both messages are treated as bit strings, she would probably not try to decrypt the message $\{N_B\}_{K_B^+}$ by using K_B^- as the decryption key. Even if she does so, she will only obtain the bit string value of N_B, i.e., $(T, E_{dy}) \models \mathbf{K}re(N_B)$. Note that, due to the lack of message meaning, Alice is not aware it is the value of N_B. Therefore, if Bob asks Alice to generate the value of N_B for him, she would have no idea how to do that. In this sense, it is unreasonable to say she "knows" (to be made clear in the next section) N_B.

2.1.3 Useful Lemmas

To this end, we enlist some helpful lemmas for future use.

Lemma 2.1.6. $T\mu \vdash t$ if and only if $T \vdash t'$ for some t' such that $t'\mu =_s t$.

Lemma 2.1.7. Suppose that all terms in $T\sigma$ are regular. If $T \vdash s$ and $s\sigma =_s C[l\theta]$, then there exists a $u \subseteq s$ such that $T \vdash u$ and $u\sigma =_s l\theta$.

Proof. The proof is by induction on $\|C\|$. If $\|C\| = 1$, then the claim is true by letting $u =_s s$. Now, we suppose the claim is true for all $\|C\| \leq k$.

For $\|C\| = k+1$, we notice that all terms in $T\sigma$ are regular. So, $s \notin T$, because otherwise $s\sigma \in T\sigma$, giving a contradiction. Using the definition of $\vdash$, we have $T \vdash sub(s)$ and $ff(s) \in \mathcal{F}^+$. Now, it is not hard to see that there is a $s_i \in sub(s)$ such that $T \vdash s_i$ and $s_i\sigma =_s C_i[l\theta]$ for some context C_i. Note that $\|C_i\| \leq k$. By induction hypothesis, there exists a $u \subseteq s_i \subset s$ such that $T \vdash u$ and $u\sigma =_s l\theta$. □

Lemma 2.1.8. Let T be a term set and t be a term.

(i). If $T \vdash t$ and $\|t\| = 1$, then $t \in T$;

(ii). If $T \vdash t$ and $T \backslash \{s\} \vdash s$, then $T \backslash \{s\} \vdash t$;

(iii). If $T \vdash t$ and $T \backslash \{s\} \nvdash t$, then $s \subseteq t$;

(iv). If $T \vdash t$, then there exists a term set $S \subseteq T$ such that $S \vdash t$ and $\|S\| \leq \|t\|$;

(v). If $T \vdash t$ and $t \notin T$, then there exists a term set $S \subseteq T$ such that $S \vdash t$ and $\|S\| < \|t\|$;

(vi). Suppose $\|s\| \geq \|t\| > 1$. $T \vdash sub(t)$ if and only if $T \backslash \{s\} \vdash sub(t)$;

(vii). Suppose that $\|t\| > 1$. $T \vdash sub(t)$ if and only if $T \backslash \{t\} \vdash sub(t)$;

Proof. (i). follows immediately from the definition of $\vdash$.

(ii). The proof is by induction on the size of t. For the base case ($\|t\| = 1$), $t \in T$ follows from (i). Then, $T \backslash \{s\} \nvdash t$ implies $s =_s t$ and thus $T \backslash \{s\} \nvdash s$, a contradiction. So, $T \backslash \{s\} \vdash t$ for $\|t\| = 1$. Now, suppose that the claim is true whenever $\|t\| \leq k$. For the induction step ($\|t\| = k+1$), if $t \in T$, as before the claim is true. Otherwise, $T \vdash t$ implies $T \vdash sub(t)$ and $ff(t) \in \mathcal{F}^+$. For every $w \in sub(t)$, we notice that $T \vdash w$. Then, by induction hypothesis, we get $T \backslash \{s\} \vdash w$ for every $w \in sub(t)$, that is $T \backslash \{s\} \vdash sub(t)$. Considering $ff(t) \in \mathcal{F}^+$, we have $T \backslash \{s\} \vdash t$, as required.

(iii). Let $T \vdash^{(n)} t$. We make induction on n. For the base case, $n = 1$, we have

$t \in T$. Suppose that the claim is true for $1 \leq n \leq k$.

For $n = k+1$, we let $t =_s f(t_1, \cdots, t_m)$. By the definition of $\vdash$, we have

$$\frac{T \vdash^{(n_1)} t_1 \cdots T \vdash^{(n_m)} t_m}{T \vdash^{(n)} f(t_1, \cdots, t_m)} \; f \in \mathcal{F}^+$$

where $n_i \leq k$ for $1 \leq i \leq m$. Since $T \backslash \{s\} \nvdash t$ and $f \in \mathcal{F}^+$, it is not hard to see that there exists a t_i $(1 \leq i \leq m)$ such that $T \backslash \{s\} \nvdash t_i$. Consider now, $T \vdash^{n_i} t_i$, $T \backslash \{s\} \nvdash t_i$, and $n_i \leq k$. By induction hypothesis, we have $s \subseteq t_i \subset t$. This completes the proof.

(iv). Let $T \vdash^{(n)} t$. We make induction on n. For the base case, $n = 1$, by the definition of $\vdash$ we have $t \in T$. Let $S = \{t\}$. We have $S \vdash t$ and $\|S\| \leq \|t\|$. So, we suppose the claim holds for $1 \leq n \leq k$.

For $n = k+1$, we let $t =_s f(t_1, \cdots, t_m)$. By the definition of $\vdash$, we have

$$\frac{T \vdash^{(n_1)} t_1 \cdots T \vdash^{(n_m)} t_m}{T \vdash^{(n)} f(t_1, \cdots, t_m)} \; f \in \mathcal{F}^+$$

where $n_i \leq k$ for $1 \leq i \leq m$. For each t_i, there is a term set $S_i \subseteq T$ such that $S_i \vdash t_i$ and $\|S_i\| \leq \|t_i\|$. Considering $f \in \mathcal{F}^+$, we have $\cup_i S_i \vdash t$. Let $S = \cup_i^m S_i$. Then,

$$\|S\| \leq \sum_i^m \|S_i\| \leq \sum_i^m \|t_i\| < \|t\|$$

This completes the proof.

(v). Since $t \notin T$, by the definition of $\vdash$, $t =_s f(t_1, \cdots, t_n)$ and

$$\frac{T \vdash^{(n_1)} t_1 \cdots T \vdash^{(n_n)} t_n}{T \vdash^{(n)} f(t_1, \cdots, t_n)} \; f \in \mathcal{F}^+$$

For each t_i, by (iv) there is a term set $S_i \subseteq T$ such that $S_i \vdash t_i$ and $\|S_i\| \leq \|t_i\|$. Considering $f \in \mathcal{F}^+$, we have $\cup_i S_i \vdash f(t_1, \cdots, t_n) =_s t$. Let $S = \cup_i^m S_i$. Then,

$$\|S\| \leq \sum_i^m \|S_i\| \leq \sum_i^m \|t_i\| < \|t\|$$

This completes the proof.

(vi). The 'if' part is trivial. We prove the 'only if' part now. Let $t =_s f(t_1, \cdots, t_n)$.

For each t_i $(1 \le i \le n)$, it follows from (iv) that there exists a term set $S_i \subseteq T$ such that $S_i \vdash t_i$ and $\|S_i\| \le \|t_i\|$. Clearly, $\cup_i S_i \vdash t_i$. Moreover,

$$\| \cup_i S_i\| \le \sum_i \|S_i\| \le \sum_i \|t_i\| < \|t\| \le \|s\|$$

So, $s \notin \cup_i S_i \subseteq T$ and thus $\cup_i S_i \subseteq T \backslash \{s\}$. Consider again $\cup_i S_i \vdash \{t_1, t_2, \cdots, t_n\}$ and $\cup_i S_i \subseteq T\backslash\{s\}$. Finally, we obtain $T\backslash\{s\} \vdash sub(t)$.

(vii). It follows immediately from (vi). $\square$

Lemma 2.1.9. Let T be a term set, t be a term, and C be a context. Suppose that u does not occur in T.

(i). If $T \vdash C[u]$ and $T \vdash v$, then $T \vdash C[v]$;

(ii). If $T \vdash t$ and $T \vdash v$, then $T \vdash t[u \mapsto v]$;

(iii). If $T \vdash C[u]$ and $T \nvdash C[v]$, then $T \vdash u$ and $T \nvdash v$;

Proof. (i). Since u does not occur in T and thus $C[u] \notin T$, we have $ff(C[u]) \in \mathcal{F}^+$ and $T \vdash sub(C[u])$ whenever $ff(C[u]) \in \mathcal{F}^+$ by the definition of $\vdash$. We make induction on the size of C.

For the base case, $\|C\| = 1$ (i.e., $C =_s \square$), $T \vdash C[v]$ is trivial. We suppose that the claim is true for $\|C\| \le k$. If $\|C\| = k+1$, $ff(C) \in \mathcal{F}^+$. Let $C[u] =_s f(t_1, \cdots, t_n)$. Then $T \vdash t_i$ for $1 \le i \le n$. It is not hard to see that there exists one $t_j \in sub(C[u])$ $(1 \le j \le n)$ such that $t_j =_s C'[u]$ for some context C' and $C[v] =_s f(t_1, \cdots, t_{j-1}, C'[v], t_{j+1}, \cdots t_n)$. Applying the induction hypothesis, we get $T \vdash C'[v]$. Consider now, $ff(C) \in \mathcal{F}^+$, $T \vdash t_i$ for $1 \le i \le n$, and $T \vdash C'[v]$, we obtain $T \vdash C[v]$. This completes the proof.

(ii). It follows from (i).

(iii). We make induction on the size of C. The base case, $\|C\| = 1$ (i.e., $C =_s \square$), is trivial. Suppose that the claim is true for $\|C\| \leq k$.

For the base case, $\|C\| = k+1$, let $C[u] =_s f(t_1, \cdots, t_n)$. By the definition of context, there exists one $t_i \in sub(C[u])$ $(1 \leq j \leq n)$ such that $t_i =_s C'[u]$ for some context C' and $C[v] =_s f(t_1, \cdots, t_{i-1}, C'[v], t_{i+1}, \cdots t_n)$. To establish $T \vdash C[u]$, there are two cases to be considered.

(*Case 1*): $C[u] \in T$. Clearly, u occurs in T, a contradiction.

(*Case 2*): $f \in \mathcal{F}^+$ and $T \vdash sub(C[u])$. Observe that $T \nvdash C[v]$. So, we have $T \nvdash C'[v]$. Consider now, $T \vdash C'[u]$, $T \nvdash C'[v]$, and $\|C'\| \leq k$. By induction hypothesis, we get $T \vdash u$ and $T \nvdash v$. $\square$

Lemma 2.1.10. If $T \vdash t$, $T \nvdash s$, and $s \subset t$, then there exists a term u such that $s \subseteq u$, $u \subseteq t$, and $u \in T$.

Proof. Clearly, $\|t\| \geq 2$. We make induction on the size of t. For $\|t\| = 2$, $s \subset t$ if and only if $s \in sub(t)$. If $t \in T$, then the claim holds by letting $u =_s t$. Otherwise, since $T \vdash t$, it follows from the definition of $\vdash$ that $T \vdash sub(t) = \{s\}$ and $ff(t) \in \mathcal{F}^+$. Moreover, since $\|s\| = 1$, we have $s \in T$ by Lemma 2.1.8 (i) and thus the claim holds by letting $u =_s s$. Now, suppose that the claim holds for $\|t\| \leq k$.

For $\|t\| = k+1$, since $s \subset t$, it is clear from the definition of $\subset$ that $s \subseteq w$ for some $w \in sub(t)$. If $t \in T$, then the claim holds by letting $u =_s s$, because $s \subset t$ implies $s \subset t$. Otherwise, as before, $T \vdash t$ implies $T \vdash sub(t)$ and $ff(t) \in \mathcal{F}^+$. Clearly, $s \neq_t w$, because $T \nvdash s$ by assumption. So, $s \subset w$. Note that $T \vdash w$, $T \nvdash s$, $s \subset w$, and $\|w\| \leq k$. By induction hypothesis, there exists a term u such that $s \subseteq u$, $u \subseteq w \subset t$, and $u \in T$. The claim follows. $\square$

2.2 The *de dicto* Interpretation

Now, we formalize the *de dicto* interpretation of a message. We first explain why *de dicto* amounts to ascribing meaning to a message and then we generalize the *de dicto* interpretation to ambiguous messages.

2.2.1 Ascribing Meaning to a Message

As mentioned earlier in the beginning of this chapter, under *de dicto* reading, knowing a message means being able to *determine* the value of it, without necessarily knowing the value. We note that the bit string value of message t is determined if t is a ground term (i.e., $t \in \mathcal{T}(\mathcal{F}, \mathcal{A} \cup \mathcal{N}, \emptyset)$). For example, the value of term N_A is $\lfloor N_A \rfloor$, where $\lfloor\rfloor$ is a unary function that maps a ground term to its bit string representation.

The *de dicto* interpretation seems trivial for ground terms, as every ground term is defined to stand for some determined (but not necessarily known) bit string value. The interpretation makes more sense when the meaning of a term is not evident, that is, a term with variable(s).

Definition 2.2.1 (Ground *de dicto* Knowledge). Every ground term t is *de dicto* known, written $\models \mathbf{K}\texttt{dicto}(t)$.

For ease of presentation, we continue to use (T, E) to model an agent's knowledge, where T is a ground term set. Under the *de re/ dicto* interpretation, the agent knows both the value and the meaning of t for each $t \in T$.

Definition 2.2.2 (Ground Knowledge). Let E be an equational theory and T be a ground term set. We say a ground term t is known in model (T, E), written $(T, E) \models \mathbf{K}t$, if $T \vdash_E t$.

For instance, in Example 1, given the meaning of each term in T, Alice is able to compute $\lfloor N_B \rfloor$ and she is aware of the fact that it is the value of term N_B. That is, $(\{\{N_B\}_{K_B^+}, K_B^-\}, E_{dy}) \models \mathbf{K} N_B$.

Remark 1. It can be feasible to compute the bit string value of a term t based on its *de dicto* interpretation. Conversely, given a bit string value, it is almost always infeasible to obtain the corresponding term structure. As an example, let $(\{N_A, K_B^+\}, E_{dy})$ be Alice's knowledge model. Then, $(\{N_A, K_B^+\}, E_{dy}) \models \mathbf{K}\mathtt{re}(\{\{\{N_A\}_{K_B^+}\}_{K_B^+}\}_{K_B^+})$. That is to say, Alice knows how to generate the bit string of term $\{\{\{N_A\}_{K_B^+}\}_{K_B^+}\}_{K_B^+}$, as long as the term structure is evident. Now, suppose that $\lfloor \{\{\{N_A\}_{K_B^+}\}_{K_B^+}\}_{K_B^+} \rfloor$ is $0x1A3DE405$. It is infeasible for Alice to get to know that the value $0x1A3DE405$ corresponds to the term $\{\{\{N_A\}_{K_B^+}\}_{K_B^+}\}_{K_B^+}$.

Example 2. Consider again the Abadi-Cortier argument. We use $\{n_0, n_1, \{v\}_{K_S^+}\}$ to describe the attacker's explicit knowledge, in which Boolean values true (n_1) and false (n_0) are "well-known". By letting

$$E_0 = E_{dy} \cup \{v = n_0\}, \quad E_1 = E_{dy} \cup \{v = n_1\}$$

we get $T \vdash_{E_0} v$ and $T \vdash_{E_1} v$. In both cases, we see from Definition 2.2.2 that $(\{n_0, n_1, \{v\}_{K_S^+}\}, E_0) \models \mathbf{K} v$ and $(\{n_0, n_1, \{v\}_{K_S^+}\}, E_1) \models \mathbf{K} v$. Consequently, the attacker knows v, giving a contradiction to our intuition.

The above contradiction occurs because the equations $v = n_0$ and $v = n_1$ are not well-established. By well-established, we mean the equation reflects some well-known fact, such as the equation $\mathtt{sdec}(\mathtt{senc}(x, y), y) = x$ used for symmetric encryption, or incorporates some initial system assumption, such as $N_B = N_A + 1$. The two new equations in Example 2, however, are introduced to model uncertainty about v.

2.2.2 Accounting For Uncertainty

Until now we have focused on ground knowledge, that is, the meaning of a term is determined. Next, we show how to use free variables and substitutions to capture uncertainty in security protocol analysis.

We use a variable to stand for an ambiguous (part of) message, and a substitution to assign one possible interpretation of the variable. For instance, we replace constant symbol v in Example 2 with a variable x to account for the uncertainty. Then, term set $\{n_0, n_1, \{x\}_{K_S^+}\}$, together with $\sigma_0 = [n_0/x]$ and $\sigma_1 = [n_1/x]$, characterizes the fact the interpretation of x is not determined yet.

In security protocol executions, a received message almost always has some part(s) being ambiguous. Let us consider again the Otway-Rees protocol [96]:

Message 1. $A \to B : M, A, B, \{N_A, M, A, B\}_{K_{AS}}$

Message 2. $B \to S : M, A, B, \{N_A, M, A, B\}_{K_{AS}}, \{N_B, M, A, B\}_{K_{BS}}$

Message 3. $S \to B : M, \{N_A, K_{AB}\}_{K_{AS}}, \{N_B, K_{AB}\}_{K_{BS}}$

Message 4. $B \to A : M, \{N_A, \underline{K_{AB}}\}_{K_{AS}}$

After executing the first three messages, principal A is expecting a K_{AB}, which is a symmetric key shared between A and B, from the trusted third party S. As K_{AB} is dynamically generated, A is uncertain about the value of it. So, A would rather use a variable x to stand for it, with substitution $\sigma = [K_{AB}/x]$ specifying its *intended* interpretation. Other interpretations of x are also possible. Particularly, the protocol is vulnerable to a type-flaw attack [28] due to the interpretation $\sigma' = [M \cdot A \cdot B/x]$.

We stress that a variable may have several interpretations (specified by substitutions). Intuitively, a term t is determined or known under the *de dicto* reading, if the agent has certain expectation about its bit string representation [57]; that is, given a bit string, though the agent may not necessarily know it under the *de re* reading, he or she can ensure that the meaning of the bit string is fixed.

2.3 Knowledge Model

So far, we have developed some on-the-fly models to reason about ground knowledge, and informally discussed knowledge with uncertainty. In this section, we propose a more general knowledge model that treats the *de re/ dicto* interpretations in a uniform way, and represents epistemic uncertainty.

2.3.1 Mapping the Kripke Structure

We start with the Kripke structures [69] which are widely used to formalize the standard possible-worlds semantics of knowledge [54]. The intuitive idea behind the possible-worlds semantics is that: due to the most likely partial observations of the actual world, an agent may not be able to know the real state of the world, but rather consider a number of other possible states that are consistent with his or her current observations. The agent knows a fact if the fact is true at all those states that he or she considers possible.

Formally, for a multi-agent system with n agents, a Kripke structure is defined as a tuple $(S, \pi, \mathcal{K}_1, \cdots, \mathcal{K}_n)$, where S is a set of *states* or *worlds*, π is an *truth assignment function*, and $\mathcal{K}_i$ is a *possibility* relation for agent i. The truth assignment function π tells the true affairs in a given state. The possibility relation $\mathcal{K}_i$ is a binary relation on S; it captures the fact that an agent i is unable to distinguish between s and t, given his information in the world s and $(s,t) \in \mathcal{K}_i$. Moreover, $\mathcal{K}_i$ is often required to be reflexive, symmetric, and transitive.

Example 3. Consider, for example, Alice is rolling two dice (say Dice A and Dice B) and the Kripke structure describing this scenario is $(S, \pi, \mathcal{K}_{Alice})$. We use an ordered pair (a, b) to denote a state, where a and b are the numbers on the top of the Dice A and Dice B, respectively. Clearly, S is the set of all the outcomes of rolling these dice, i.e.,

$$S = \{(a, b) | 1 \leq a \leq 6, 1 \leq b \leq 6\}$$

Suppose that Alice does not directly know of the outcome of the two dice, but rather only observe the sum of the two dice. So, we use proposition p_i to denote the fact "the sum of the two dice is equal to i". Clearly, $2 \leq i \leq 12$.

For the assignment function π, we have

$$\pi((a, b))(p_i) \triangleq \begin{cases} \textbf{true} & \text{if } a + b = i \\ \\ \textbf{false} & \text{if } a + b \neq i \end{cases}$$

For the possibility relation for Alice $\mathcal{K}_{Alice}$, we have

$$((a, b), (a', b')) \in \mathcal{K}_{Alice} \Leftrightarrow a + b = a' + b'$$

Assume that the actual state is $(2, 3)$, and yet the only observation Alice has is "p_5: the sum of the two dice is equal to 5". Then, Alice considers all the following states are possible or indistinguishable:

$$(1, 4), (2, 3), (3, 2), (4, 1)$$

To adopt a Kripke-style structure for our purpose, we make the following changes. (i). As explained in the beginning of this chapter, it suffices to consider only a single agent, and thus we will keep only one possibility relation $\mathcal{K}$; (ii). There is no need to include the truth assignment function π, because the truth value of a statement on a message is implicitly determined by one's initial knowledge and reasoning capabilities. Rather, we need a term set T and an equational theory E built into the structure so as to represent the agent's initial knowledge and reasoning capabilities (i.e., deducibility). (iii). In our context, a state can be fully characterized by a substitution σ. We thus use a nonempty set of substitutions or *valid domain* Φ to denote a set of states that are of interest. Note that Φ is nonempty even if it only contains an empty substitution ϵ (i.e., $\Phi = \{\epsilon\}$). In Section 3.2, we will see how we can get rid of possibility relation $\mathcal{K}$ and the set of interested substitution Φ. For now, let us keep the model in its most general form.

Finally, we define *knowledge model* or *knowledge structure* as a tuple $M = (E, T, \Phi, \mathcal{K})$, where E is an equational theory, T is a term set, Φ is a set of substitutions satisfying that $Dom(\sigma) \subseteq fv(T)$ for all $\sigma \in \Phi$, and $\mathcal{K}$ is a possibility relation. We often refer to the term set T in M as *explicit knowledge*, as it is used exactly to represent the agent's explicit knowledge. Similarly, the substitution set Φ in M is also called *valid domain*, as it describes all substitutions that are of interest. A *knowledge state* is a pair (M, σ), where M is a knowledge model with valid domain Φ and $\sigma \in \Phi$.

A valid domain identifies a set of substitutions that are possible in a specific problem domain. For example, if we use x to represent a user password and Φ the valid domain, then $x\sigma$ is a possible user password for every $\sigma \in \Phi$. Here, we use the term "possible" in a sense that the value of $x\sigma$ satisfies some preconditions on passwords, such as the length of a password should be no less than 6, a password can not be the same as one's username, and etc. To avoid confusion with the possibility relation $\mathcal{K}$, we use the term "valid" instead.

It should be noted that, to simplify our discussion we have not included the term algebra used $\mathcal{T}$ in M, but rather make it an assumption.

2.3.2 Modeling Knowledge

We now formalize the meaning of knowing a message under different interpretations.

Definition 2.3.1 (Knowledge Model). Given a knowledge model $M = (E, T, \Phi, \mathcal{K})$ and $\sigma \in \Phi$, we define $\models$ as follows:

(i). $(M, \sigma) \models \mathbf{K}\texttt{re}(s)$ if and only if $T\sigma \vdash_E s$,

(ii). $(M, \sigma) \models \mathbf{K}\texttt{dicto}(t)$ if and only if $fv(t) \subseteq fv(T)$ and $t\sigma' =_E t\sigma$ for all σ' such that $\sigma' \in \Phi$ and $(\sigma, \sigma') \in \mathcal{K}$,

(iii). $(M, \sigma) \models \mathbf{K}t$ if and only if $(M, \sigma) \models \mathbf{K}\texttt{re}(t\sigma)$ and $(M, \sigma) \models K\texttt{dicto}(t)$,

We write $M \models \mathbf{K}t$ if and only if $(M, \sigma) \models \mathbf{K}t$ for every $\sigma \in \Phi$.

An agent knows a term if and only if the bit string value can be computed and the meaning of the term is determined. If T is ground (i.e., $\Phi = \{\epsilon\}$ and $\mathcal{K}$=$\emptyset$), the above definition reduces to Definition 2.1.5 and 2.2.1. This suggests that deducibility is a notion sufficient to capture agent's knowledge if we do not have the need to reason about uncertainty.

At this point, we provide a precise answer to the question "what is meant by saying the one knows some message?" That is, an agent *knows* a message t if and only if $(M, \sigma) \models \mathbf{K}t$, where (M, σ) models the agent's knowledge state.

In the following, we collect two examples to illustrate the use of the knowledge model. As reasoning about ground knowledge reduces to the well studied notion of deducibility, here we focus on knowledge with uncertainty.

Example 4. Consider again the Abadi-Cortier argument. Let $M = (E_{dy}, T, \Phi, \mathcal{K})$ model the attacker's knowledge, where $T = \{n_0, n_1, \{x\}_{K_S^+}\}$, $\Phi = \{\sigma_0, \sigma_1\}$, and $\sigma_i = [n_i/x]$ for $i = 0, 1$. It is not hard to see that the attacker is unable to distinguish n_0 from n_1, so $\mathcal{K} = (\sigma_0, \sigma_1)$.

Since $T\sigma_0 \vdash_{E_{dy}} x\sigma_0 (=_s n_0)$, by Definition 2.3.1 (i) we have $(M, \sigma_0) \models \mathbf{K}\mathtt{re}(x\sigma_0)$. Likewise, $(M, \sigma_1) \models \mathbf{K}\mathtt{re}(x\sigma_1)$. Moreover, $x\sigma_0 \neq_{E_{dy}} x\sigma_1$. It follows Definition 2.3.1 (ii) that $(M, \sigma_0) \not\models \mathbf{K}\mathtt{dicto}(x)$ and $(M, \sigma_1) \not\models \mathbf{K}\mathtt{dicto}(x)$.

Finally, by Definition 2.3.1 (iii), we obtain $M \not\models \mathbf{K}x$. This corresponds to the fact that the attacker does not know the vote x.

Example 5. To continue the previous example, we slightly change the scenario by assuming that the attacker manages to eavesdrop the public key K_S^+. Then, the attacker's knowledge model becomes $M' = (E_{dy}, T', \Phi, \mathcal{K}')$, where $T' = T \cup \{K_S^+\} = \{n_0, n_1, K_S^+, \{x\}_{K_S^+}\}$.

As before, since $T'\sigma_0 \vdash_{E_{dy}} x\sigma_0 (=_s n_0)$, by Definition 2.3.1 (i) we have $(M', \sigma_0) \models \mathbf{K}\mathtt{re}(x\sigma_0)$. Likewise, $(M', \sigma_1) \models \mathbf{K}\mathtt{re}(x\sigma_1)$.

Let $u =_s \{n_0\}_{K_S^+}$ and $v =_s \{x\}_{K_S^+}$. Since $T' \vdash_{E_{dy}} u$ and $T' \vdash_{E_{dy}} v$, the attacker is able to duduce both u and v. Moreover, notice that $u\sigma_0 =_{E_{dy}} v\sigma_0$ and yet $u\sigma_1 \neq_{E_{dy}} v\sigma_1$. So, if the attacker is in state σ_0, he would observe that u is equal to v; on the other hand, if he is in state σ_1, he would notice that u is not equal to v. In other words, the attacker is able to distinguish state σ_0 from state σ_1. Therefore, we have $\mathcal{K} = \emptyset$. Then, it follows from Definition 2.3.1 (ii) that $(M', \sigma_0) \not\models \mathbf{K}\mathtt{dicto}(x)$ and $(M', \sigma_1) \not\models \mathbf{K}\mathtt{dicto}(x)$.

Altogether, by Definition 2.3.1 (iii), we have $M \models \mathbf{K}x$. This means that the privacy of x is breached.

CHAPTER 3: DEFINING RECOGNIZABILITY

The purpose of this chapter is to provide a formal treatment of verifying messages by introducing the notion of recognizability. Most of the results presented in this chapter are reported in our previous paper [72].

3.1 General Definition

As we have seen in the last chapter, without uncertainty, one's knowledge can be captured by a ground term set. In this chapter, we will see that uncertainty is at the root of "verifying" a message. Any message to be verified should be regarded as an ambiguous message.

Before proceeding any further with our general discussion, let us start with a simple example. Assume that Alice's knowledge is modeled by

$$M = (E_{dy}, \{\{N_B\}_{K_B^+}, K_B^+\}, \{\epsilon\}, \{(\epsilon, \epsilon)\})$$

It is not hard to see that Alice knows $\{N_B\}_{K_B^+}$ and K_B^+, or more formally, $(M, \epsilon) \models \mathbf{K}\{N_B\}_{K_B^+}$ and $(M, \epsilon) \models \mathbf{K}K_B^+$. Suppose that Bob sends Alice a message and tells her that the message is the nonce N_B. Since Alice does not know N_B (i.e., $(M, \epsilon) \not\models \mathbf{K}N_B$), in order to achieve certainty she has to "verify" the incoming message.

We stress that, although the incoming message is potentially ambiguous, it does affect Alice's knowledge. More specifically, Alice's knowledge model becomes

$$M' = (E_{dy}, \{\{N_B\}_{K_B^+}, K_B^+, x\}, \Phi, \mathcal{K})$$

where x stands for the message received by Alice. Without any prior information about x, we let the valid domain $\Phi = \{\sigma | Dom(\sigma) \subseteq \{x\}\}$, in which the expected state is $\sigma_e = [N_B/x] \in \Phi$. Since Alice knows $\{N_B\}_{K_B^+}$ and K_B^+, she is able to encrypt the incoming message with K_B^+ and then compare the result $\{x\sigma\}_{K_B^+}$ with $\{N_B\}_{K_B^+}$. More specifically, according to Alice the following condition holds

$$\texttt{penc}(x\sigma_e, K_B^+) =_{E_{dy}} \{N_B\}_{K_B^+} \tag{1}$$

Note that the possibility relation $\mathcal{K}$ describes the agent's inability to distinguish two states. Thus, we have

$$\texttt{penc}(x\sigma, K_B^+) =_{E_{dy}} \{N_B\}_{K_B^+} \tag{2}$$

for all σ such that $(\sigma_e, \sigma) \in \mathcal{K}$. Due to the perfect cryptography assumption, equation (2) holds only if $x\sigma =_{E_{dy}} N_B$. Therefore, $\sigma = \sigma_e$ and $\mathcal{K} = \{(\epsilon, \epsilon), (\sigma_e, \sigma_e)\}$. At this point, we see that Alice is able to verify N_B without necessarily knowing N_B (i.e., $(M', \sigma_e) \not\models \mathbf{K}N_B$). Intuitively, verifying a message is weaker than knowing a message.

Let us take a closer look at the example. The only state σ satisfying that $\sigma \in \Phi$ and $(\sigma_e, \sigma) \in \mathcal{K}$ is σ_e. So, by Definition 2.3.1 (ii), we obtain $(M', \sigma_e) \models \mathbf{K}\texttt{dicto}(x)$. In fact, one is able to verify a message if and only if the variable standing for the message is known under the *de dicto* interpretation.

Definition 3.1.1 (General Recognizability). Let (M, σ) be one's knowledge state and suppose that the knowledge state is updated to (M', σ') after receiving an ambiguous message t (denoted by z). Then, we say that t is *recognizable* by (M, σ) and write $(M, \sigma) \rhd t$, if and only if $(M', \sigma') \models \mathbf{K}\texttt{dicto}(z)$.

Let (M, σ) be the knowledge state of an agent A. We say that a message t can be "verified" by A, or A *recognizes* message t, if and only if t is recognizable by (M, σ)

(i.e., $(M, \sigma) \rhd t$). This gives a precise answer to the question in the introduction. Here and after, we avoid the vague term "verify", but rather use "recognize" in its precise meaning necessary for rigorous protocol analysis.

Proposition 3.1.2. Let (M, σ) be an agent's knowledge state and suppose that the knowledge state is updated to (M', σ') after receiving an ambiguous message t (denoted by z). If $(M, \sigma) \models \mathbf{K}t$, then $(M, \sigma) \rhd t$.

The above proposition recovers the intuition: if one knows a message (under both the *de re* and *de dicto* readings), then he or she certainly can "verify" the message, but not vice versa. In other words, one may be able to "verify" a message without necessarily knowing the message. We will see several examples that confirm this point throughout the thesis.

Definition 3.1.1 of recognizability, though general enough, is not practically useful, as it is far from clear how to update a knowledge state to reflect the potentially ambiguous message. The next two sections deal with knowledge update and simplifying our general knowledge model. A revised definition of recognizability will be given in Section 3.5.

3.2 Knowledge Update

As we have seen in Definition 3.1.1, knowledge update is at the root of the notion of recognizability. In this section, we discuss how an agent updates the knowledge state when he or she receives a new message.

Without loss of generality, assume the initial knowledge state of an agent is (M_0, σ_0), where $M_0 = (E_0, T_0, \Phi_0, \mathcal{K}_0)$, and a new incoming message is intended to be term t.

After receiving the new message, let us assume the new knowledge state is (M', σ'), where $M' = (E', T', \Phi', \mathcal{K}')$. Before performing any internal checks on the incoming message, the agent is uncertain about the new message; even if the new message is indeed in the agent's explicit knowledge T_0, without comparing it with what is explicitly known (i.e., T_0) the agent is still unable to gain certainty about the message. So, any incoming message should be treated as an ambiguous message in the first place. That said, the agent's explicit knowledge T_0 is updated to $T' = T_0 \cup \{x\}$, where x is a fresh free variable used to denote the incoming message.

Since equational theory E_0 in the knowledge model is used to capture the underlying algebraic properties of security primitives used in the protocol and deducibility, it is independent of one's explicit knowledge. Despite the updated explicit knowledge T', the equational theory E' in the new knowledge model M' remains the same (i.e., $E' = E_0$).

For the interested domain Φ', since a new variable x is introduced to the agent's explicit knowledge T' and, by the definition of knowledge model, $dom(\sigma) \subseteq fv(T') = fv(T) \cup \{x\}$ for every $\sigma \in \Phi'$, Φ' is usually a superset of Φ. A common way to update the interested domain is simply to expand Φ to include all valid evaluations of x. Given two sets of substitutions Φ_1 and Φ_2 such that $Dom(\Phi_1) \cap Dom(\Phi_2) = \emptyset$, we define $\Phi_1 \bowtie \Phi_2$ by $\Phi_1 \bowtie \Phi_2 = \{\sigma | \sigma = \sigma_1\sigma_2 \text{ where } \sigma_1 \in \Phi_1 \text{ and } \sigma_2 \in \Phi_2\}$. Suppose that all valid evaluations of x is Φ_x ($dom(\sigma) \subseteq \{x\}$ for all $\sigma \in \Phi_x$). Then, a most common way to update Φ_0 is by letting $\Phi' = \Phi_0 \bowtie \Phi_x$.

Unlike the equational theory which remains the same regardless of the agent's explicit knowledge, the possibility relation updates as the explicit knowledge evolves.

This is because the agent's ability to distinguish message relies crucially on the agent's explicit knowledge and the interested domain as well. Since both the explicit knowledge and the valid domain change in knowledge update, the possibility relation should change accordingly. While the agent's is gaining new information, we have $\mathcal{K}' \subseteq \mathcal{K}_0$ for the new possibility relation $\mathcal{K}'$. In other words, the added information gives the agent more power to differentiate two messages. But still, it is nontrivial to derive $\mathcal{K}'$ directly from T' and Φ' and that is probably the reason why $\mathcal{K}$ is often given in general knowledge reasoning problems.

At this point, we see that the problem of updating knowledge boils down to the problem of updating the valid domain Φ and the possibility relation $\mathcal{K}$, which is nontrivial in general. Moreover, keeping both Φ and $\mathcal{K}$ in the agent's knowledge model complicates the knowledge reasoning tasks. In fact, as we shall see in the next section, it is possible to get rid of both components in the knowledge model in formal security protocol analysis.

3.3 Operational Equivalence

Indeed, it is generally hard, if not impossible, to define the possibility relation in terms of the explicit knowledge and the valid domain. Nonetheless, as we have limited our scope only to security protocols, there is still hope for a more convenient way to characterize the possibility relation.

In fact, in Example 5 we have already seen how to use the term set T and the equational theory E in the knowledge structure to derive the possibility relation $\mathcal{K}$. The rationale is that an agent's ability or inability can be fully characterized by his current information and his reasoning capabilities, which are represented by T and E, respectively. As the discussion in Example 5 is rather ad hoc, in this section we will see how to rigorously define the possibility relation $\mathcal{K}$ based on a term set T and an equational theory E.

In possible-worlds semantics, the possibility relation captures an agent's inability to distinguish two different worlds/states. In view of Definition 2.3.1, what an agent knows is all about messages. So, checking equality of messages is probably the only way to distinguish different states (i.e., substitutions). Informally, two states are indistinguishable, namely $(\sigma_1, \sigma_2) \in \mathcal{K}$, if different computations that output the same bit string in state σ_1 also output the same bit string in state σ_2, and vice versa. This suggests the following definition.

Definition 3.3.1 (Operational Equivalence). Let E be an equational theory, T be a term set, and σ_1 and σ_2 be two substitutions such that $Dom(\sigma_1) \subseteq fv(T)$ and $Dom(\sigma_2) \subseteq fv(T)$. We say that σ_1 and σ_2 are *operationally equivalent* in equational theory E with respect to 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 $u\sigma_1 =_E v\sigma_1 \Leftrightarrow u\sigma_2 =_E v\sigma_2$.

The above definition captures the fact in security executions a protocol participant can differentiate two messages only by equality checks. We thus define $\mathcal{K}_{(E,T,\Phi)}$ as follows: $(\sigma_1, \sigma_2) \in \mathcal{K}_{(E,T,\Phi)}$ if and only if $\sigma_1 \approx_{E,T} \sigma_2$ and $\{\sigma_1, \sigma_2\} \subseteq \Phi$, where E is an equational theory E, T a term set T, and Φ a substitution set.

It should be noticed that operational equivalence is closely related to static equivalence [4, 3]. The main difference is that operational equivalence is from a *cognitive* perspective, whereas static equivalence is from a *process* point of view. Moreover, deciding recognizability and deciding static equivalence are significantly different. For recognizability, we concern with the problem: given a message m whether there exists another message m' that is indistinguishable from m by the observer. In other words, we need to consider all possible message m' that is relevant to the operational equivalence relation. Consequently, deciding recognizability can be much harder than deciding static equivalence. We defer the problem of deciding recognizability to the next chapter.

Example 6. Consider the term set $T = \{N_A, K_B^-, x\}$ and let

$$\begin{aligned}
\sigma_1 &= [\{N_A \cdot A\}_{K_B^+}/x] \\
\sigma_2 &= [\{N_A \cdot \{N_B\}_{K_A^+}\}_{K_B^+}/x] \\
u &=_s \mathtt{fst}(\mathtt{pdec}(x, K_B^-)) \\
v &=_s N_A
\end{aligned}$$

Clearly, $T \vdash \{u, v\}$. We see that

$$\begin{aligned}
u\sigma_1 &=_s \mathtt{fst}(\mathtt{pdec}(\{N_A \cdot A\}_{K_B^+}, K_B^-)) \\
&\rightarrow_{R_{E_{dy}}} \mathtt{fst}(N_A \cdot A) =_s N_A =_s v\sigma_1
\end{aligned}$$

and

$$\begin{aligned}
u\sigma_2 &=_s \mathtt{fst}(\mathtt{pdec}(\{N_A \cdot \{N_B\}_{K_A^+}\}_{K_B^+}, K_B^-)) \\
&\rightarrow_{R_{E_{dy}}} \mathtt{fst}(N_A \cdot \{N_B\}_{K_A^+}) =_s N_A =_s v\sigma_2
\end{aligned}$$

So, $u\sigma_1 =_{E_{dy}} v\sigma_1$ and $u\sigma_2 =_{E_{dy}} v\sigma_2$. It can be shown that for any u and v such that $T \vdash \{u, v\}$ we have $u\sigma_1 =_{E_{dy}} v\sigma_1 \Leftrightarrow u\sigma_2 =_{E_{dy}} v\sigma_2$. That is, $\sigma_1 \approx_{E_{dy},T} \sigma_2$.

This example illustrates how a message or part of the message could be type-flawed. In fact, $\sigma_1 \approx_{E,T} \sigma$ for any substitution σ satisfying $x\sigma =_s N_A \cdot t$ where t is an arbitrary ground and R_E-normal term. This is not surprising, because if one explicitly knows N_A (i.e., $T \vdash_E N_A$), then any message part t that represents N_A (i.e., $\lfloor t \rfloor$) could be recognized by simply comparing $\lfloor t \rfloor$ and $\lfloor N_A \rfloor$.

On the other hand, if one does not explicitly know a message, will he or she still be able to verify the message? The answer depends on what exactly one knows and what the unknown message is or expected to be. In the following example, the answer is positive.

Example 7. Consider the term set $T = \{K_B^+, \{N_A\}_{K_B^+}, x\}$ and let

$$\begin{array}{ll} \sigma_1 & =[K_B^-/x] \\ u & =_s \texttt{penc}(\texttt{pdec}(\{N_A\}_{K_B^+}, x), K_B^+) \\ v & =_s \{N_A\}_{K_B^+} \end{array}$$

Clearly, $T \vdash \{u, v\}$. We see that

$$\begin{aligned} u\sigma_1 &=_s \texttt{penc}(\texttt{pdec}(\{N_A\}_{K_B^+}, K_B^-), K_B^+) \\ &=_s \texttt{penc}(N_A, K_B^+) =_s v =_s v\sigma_1 \end{aligned}$$

So, $u\sigma_1 =_{E_{dy}} v\sigma_1$. Assume that $\sigma_1' \approx_{E_{dy},T} \sigma_1$. Then, $u\sigma_1' =_{E_{dy}} v\sigma_1'$. That is,

$$\texttt{penc}(\texttt{pdec}(\{N_A\}_{K_B^+}, x\sigma_1'), K_B^+) =_{E_{dy}} v\sigma_1' =_s \{N_A\}_{K_B^+}$$

Now, it is not hard to see that $\texttt{pdec}(\{N_A\}_{K_B^+}, x\sigma_1') =_s N_A$ and thus $x\sigma_1' =_s K_B^-$. Note that $Dom(\sigma_1') = Dom(\sigma_1) = \{x\}$. Finally, we get $\sigma_1' = [K_B^-/x] = \sigma_1$.

Similarly, if we let $\sigma_2 = [N_A/x]$, it can be shown that $\sigma_2 \approx_{E_{dy},T} \sigma_2'$ if and only if $\sigma_2' = \sigma_2$.

In the above example, although neither N_A nor K_B^- is explicitly known ($T \nvdash_E \{N_A, K_B^-\}$), one can still verify them, because for any $\sigma_1' \approx_{E,T} \sigma_1$ and $\sigma_1' \approx_{E,T} \sigma_1$ we have $\sigma_1' = \sigma_1$ and $\sigma_2' = \sigma_2$.

The following lemma and theorem give some useful characterizations of operational equivalence.

Lemma 3.3.2. Let σ_1 and σ_2 be two ground substitutions.

(i). $\sigma_1 \approx_{E,T} \sigma_2$ if and only if $\sigma_2 \approx_{E,T} \sigma_1$;

(ii). if $\mu\sigma_1 \approx_{E,T} \mu\sigma_2$ and $Dom(\sigma_1) = fv(T\mu)$, then $\sigma_1 \approx_{E,T\mu} \sigma_2$;

Proof. (i). Follows immediately from Definition 3.3.1.

(ii). Without loss of generality, let u and v be two terms such that $T\mu \vdash \{u, v\}$. By Lemma 2.1.6, there exists two terms u' and v' such that $T \vdash \{u', v'\}$, $u'\mu =_s u$, and $v'\mu =_s v$. Moreover, Since $\mu\sigma_1 \approx_{E,T} \mu\sigma_2$ and $T \vdash \{u', v'\}$, we have $u'\mu\sigma_1 =_E v'\mu\sigma_1 \Leftrightarrow u'\mu\sigma_2 =_E v'\mu\sigma_2$. That is, $u\sigma_1 =_E v\sigma_1 \Leftrightarrow u\sigma_2 =_E v\sigma_2$. Moreover, $Dom(\sigma_1) = fv(T\mu)$ by assumption. Using the definition of operational equivalence, we know that $\sigma_1 \approx_{E,T\mu} \sigma_2$. □

Theorem 3.3.3. Let σ_1 and σ_2 be two ground substitutions.

(i). Suppose that $T \vdash_E t$. Then, $\sigma_1 \approx_{E,T} \sigma_2$ if and only if $\sigma_1 \approx_{E,T\cup\{t\}} \sigma_2$;

(ii). Suppose that $T \vdash t$ and x never occurs in T. Let $t\sigma_1 \to^!_{R_E} w_1$ and $t\sigma_2 \to^!_{R_E} w_2$. Then, $\sigma_1 \approx_{E,T} \sigma_2$ if and only if $\sigma_1' \approx_{E,T\cup\{x\}} \sigma_2'$, where $\sigma_1' = \sigma_1 \cup [w_1/x]$ and $\sigma_2' = \sigma_2 \cup [w_2/x]$.

Proof. (i). The "if" part is trivial. We now prove the "only if" part. To prove $\sigma_1 \approx_{E,T\cup\{t\}} \sigma_2$, it suffices to show that for all terms u and v such that $T \cup \{t\} \vdash \{u, v\}$ we have $u\sigma_1 =_E v\sigma_1 \Leftrightarrow u\sigma_2 =_E v\sigma_2$. Due to the symmetry of σ_1 and σ_2, we only need to prove one direction and proof of the reverse direction can be easily obtained by a similar analysis.

Since $T \vdash_E t$, it is obvious that $T \cup \{t\} \equiv_E T$. Note that $T \vdash \{t\} \vdash \{u, v\}$. By the definition of $\vdash_E$, there exists two terms u' and v' such that $T \vdash \{u', v'\}$, $u' =_E u$, and $v' =_E v$. Clearly, $u\sigma_1 =_E u'\sigma_1$ and $v\sigma_1 =_E v'\sigma_1$. So, $u\sigma_1 =_E v\sigma_1$ implies $u'\sigma_1 =_E v'\sigma_1$. Note that $T \vdash \{u', v'\}$ and $\sigma_1 \approx_{E,T} \sigma_2$. By the definition of operational equivalence, we have $u'\sigma_2 =_E v'\sigma_2$ and thus $u\sigma_2 =_E v\sigma_2$. Likewise, it can be shown that $u\sigma_2 =_E v\sigma_2 \Leftrightarrow u\sigma_1 =_E v\sigma_1$. Hence, $\sigma_1 \approx_{E,T\cup\{t\}} \sigma_2$.

(ii). ("If " part) To prove $\sigma_1 \approx_{E,T} \sigma_2$, it suffices to show that for all terms u and v such that $T \vdash \{u, v\}$ we have $u\sigma_1 =_E v\sigma_1 \Leftrightarrow u\sigma_2 =_E v\sigma_2$. Clearly, $T \cup \{x\} \vdash \{u, v\}$. Since $\sigma_1' \approx_{E,T\cup\{x\}} \sigma_2'$ by assumption, we have $u\sigma_1' =_E v\sigma_1' \Leftrightarrow u\sigma_2' =_E v\sigma_2'$. Note that $T \vdash \{u, v\}$ and x does not occur in T. Obviously, $x \notin fv(u)$ and $x \notin fv(v)$. So, $u\sigma_1' =_s u\sigma_1$, $v\sigma_1' =_s v\sigma_1$, $u\sigma_2' =_s u\sigma_2$, and $v\sigma_2' =_s v\sigma_2$. Therefore, $u\sigma_1 =_E v\sigma_1 \Leftrightarrow u\sigma_2 =_E v\sigma_2$.

("Only if" part) To prove $\sigma_1' \approx_{E,T\cup\{x\}} \sigma_2'$, it suffices to show that for all terms u and v such that $T \cup \{x\} \vdash \{u, v\}$ we have $u\sigma_1' =_E v\sigma_1' \Leftrightarrow u\sigma_2' =_E v\sigma_2'$.

Let $u' =_s u[x \mapsto t]$ and $v' =_s v[x \mapsto t]$. Since x never occurs in T and $T \vdash t$ by assumption, we have $T \vdash \{u', v'\}$. Note that $\sigma_1' = \sigma_1 \cup [w_1/x]$ and $\sigma_2' = \sigma_2 \cup [w_2/x]$. It is not hard to see that $u'\sigma_1 =_s u\sigma_1[x \mapsto t\sigma_1] =_E u\sigma_1[x \mapsto w_1] =_s u\sigma_1'$. So, $u\sigma_1' =_E u'\sigma_1$. Similarly, we have $v\sigma_1' =_E v'\sigma_1$, $u\sigma_2' =_E u'\sigma_2$, and $v\sigma_2' =_E v'\sigma_2$. On the other hand, since $\sigma_1 \approx_{E,T} \sigma_2$ and $T \vdash \{u', v'\}$, by the definition of operational equivalence we get $u'\sigma_1 =_E v'\sigma_1 \Leftrightarrow u'\sigma_2 =_E v'\sigma_2$. That is, $u\sigma_1' =_E v\sigma_1' \Leftrightarrow u\sigma_2' =_E v\sigma_2'$. This completes the proof. $\square$

3.4 Knowledge Model Revised

In the last section, we have used equational theory E, explicit knowledge T, and valid domain Φ to characterize the possibility relation $\mathcal{K}_{(E,T,\Phi)}$. To make knowledge reasoning more effective in security protocol analysis, this section aims to eliminate both valid domain Φ and possibility relation $\mathcal{K}$ from the knowledge model as defined in Section 2.3.1.

In example 4, we have used free variable x to stand for a possible vote (i.e., a Boolean value). So, $x\sigma$ is a well-known value (either true or false) for every $\sigma \in \Phi$ where Φ is the valid domain in the example. More often, however, we use substitutions to represent values that are not well-known, such as a 128-bit block cipher. Enumerating all those values are intractable and unnecessary. We thus define Φ_T by $\Phi_T = \{\sigma | Dom(\sigma) \subseteq fv(T)\}$.

For an ambiguous message that has a large number of valid values, it is practicable to use Φ_T as the valid domain. In the rest of this thesis, we avoid ambiguous messages that have well-known values, but rather assume all ambiguous messages have a large number of valid values. Moreover, we assume a uniform underlying distribution of valid values; this is not true in reality, because for instance user tend to choose weak passwords with low entropy [16, 17, 102, 67]. Despite the above assumptions, we claim that these assumptions simplify the proceeding discussion, without affecting our main results of the thesis.

When Φ is defined by Φ_T, $\mathcal{K}$ is defined by $\mathcal{K}_{(E,T)}$, and equational theory E is given, one's knowledge state can be fully captured T and σ.

Definition 3.4.1 (Succinct Knowledge State). Let E be an equational theory, T be a term set, and σ be a substitution satisfying that $Dom(\sigma) \subseteq fv(T)$. Then, we define a triple $\langle E, T, \sigma\rangle$ as (M, σ), where $M = (E, T, \Phi_T, \mathcal{K}_{(E,T)})$; such a triple is called a *(succinct) knowledge state* and is notated as $\vec{T}$. We will write $\vec{T}\downarrow_{ts}$ and $\vec{T}\downarrow_{subs}$ for the term set and substitution in $\vec{T}$, respectively.

For simplicity, we will drop the equational theory E and simply write $\langle T, \sigma\rangle$, when E is clear from the context. Likewise, the knowledge model as defined in Definition 2.3.1 is revised accordingly.

Definition 3.4.2 (Succinct Knowledge Model). Given an equational theory E, we define $\models$ as follows:

(i). $\langle E, T, \sigma\rangle \models \mathbf{K}\mathtt{re}(s)$ if and only if $T\sigma \vdash_E s$,

(ii). $\langle E, T, \sigma\rangle \models \mathbf{K}\mathtt{dicto}(t)$ if and only if $fv(t) \subseteq fv(T)$ and $t\sigma' =_E t\sigma$ for all σ' such that $\sigma' \approx_{E,T} \sigma$,

(iii). $\langle E, T, \sigma\rangle \models \mathbf{K}t$ if and only if $\langle E, T, \sigma\rangle \models \mathbf{K}\mathtt{re}(t\sigma)$ and $\langle E, T, \sigma\rangle \models K\mathtt{dicto}(t)$,

In the rest of this thesis, unless stated otherwise, we only consider succinct knowledge state/model, or simply, knowledge state/model.

Example 4 shows a case when $(M, \sigma) \models \mathbf{K}\mathtt{re}(t\sigma)$ but $(M, \sigma) \not\models \mathbf{K}\mathtt{dicto}(t)$. The following example shows a reverse case, that is, $(M, \sigma) \models \mathbf{K}\mathtt{dicto}(t)$ but $(M, \sigma) \not\models \mathbf{K}\mathtt{re}(t\sigma)$.

Example 8. Let $\langle E_{dy}, T, \sigma_0\rangle$ be Alice's knowledge state, where

$$T = \{\{K_S^+, \{\{m\}_{K_B^+}\}_{K_S^+}, \{x\}_{K_B^+}\}\}$$

and $\sigma_0 = [m/x]$. Then,

$$T\sigma_0 = \{\{K_S^+, \{\{m\}_{K_B^+}\}_{K_S^+}, \{m\}_{K_B^+}\}\}$$

Without the decryption key K_B^-, Alice is unable to deduce m (i.e., $T\sigma_0 \nvdash_{E_{dy}} m$). By Definition 3.4.2 (i), we have $\langle E_{dy}, T, \sigma_0\rangle \not\models \mathbf{K}\mathtt{re}(x\sigma_0)$.

In the following, we let σ be an arbitrary substitution satisfying that $\sigma \approx_{E_{dy},T} \sigma_0$, $u =_s \{\{x\}_{K_B^+}\}_{K_S^+}$, and $v =_s \{\{m\}_{K_B^+}\}_{K_S^+}$. Since $T \vdash \{u, v\}$ and $u\sigma_0 =_{E_{dy}} v\sigma_0$, it follows from Definition 3.5.1 that $u\sigma =_{E_{dy}} v\sigma$. That is,

$$\{\{x\}_{K_B^+}\}_{K_S^+}\sigma =_{E_{dy}} \{\{m\}_{K_B^+}\}_{K_S^+}$$

Note that σ is $R_{E_{dy}}$-normal and E_{dy} is a convergent theory. Thus,

$$\{\{x\}_{K_B^+}\}_{K_S^+}\sigma =_s \{\{m\}_{K_B^+}\}_{K_S^+}$$

So, $\sigma = [m/x] = \sigma_0$. Consider now, $fv(x) \subseteq fv(T)$, $x\sigma =_{E_{dy}} x\sigma_0$ for all σ such that $\sigma \approx_{E_{dy},T} \sigma_0$. By Definition 3.4.2 (ii), $\langle E_{dy}, T, \sigma_0\rangle \models \mathbf{K}\mathtt{dicto}(x)$.

The last example shows how our knowledge model facilitates reasoning about off-line guessing attack. We will discuss more on this topic in Chapter 7.1.

Example 9. We consider a simple one-way authentication protocol:

$$\begin{array}{ll} \text{Message 1.} & A \rightarrow B : \{N_A\}_{K_{AB}} \\ \text{Message 2.} & B \rightarrow A : \{N_A + 1\}_{K_{AB}} \end{array}$$

In order to model this protocol, we slightly enrich the equational theory E_{dy}[2] with a binary function "+" to handle addition operations.

The attacker eavesdrops the communications between A and B, and aims to guess the symmetric key K_{AB}. Then, the attacker's knowledge state before making a guess of K_{AB} is $\vec{T_0} = \langle E_{dy+}, T_0, \epsilon\rangle$, where $T_0 = \{1, \{N_A\}_{K_{AB}}, \{N_A + 1\}_{K_{AB}}\}$. Without any guessed value, the initial knowledge state $\vec{T_0}$ does not reflect any uncertainty.

[2] In fact, the equational theory remains the same; only the underlying term algebra is changed to accommodate addition operations used in the one-way authentication protocol.

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

Figure 2: Equational Theory E_{dy+}.

After the attacker makes a random guess of the symmetric key K_{AB}, the knowledge state is updated to $\vec{T} = \langle E_{dy+}, T, \sigma \rangle$, where

$$T = T_0 \cup \{x\}$$

$$= \{1, \{N_A\}_{K_{AB}}, \{N_A + 1\}_{K_{AB}}, x\}$$

$$\sigma_0 = [K_{AB}/x]$$

Since a guessed value of K_{AB} can be wrong, we treat it as an ambiguous message and thus use the free variable x to stand for it.

Consider now,

$$T_0\epsilon = T_0 = \{1, \{N_A\}_{K_{AB}}, \{N_A + 1\}_{K_{AB}}\}$$

Clearly, the attacker is unable to deduce K_{AB} (i.e., $T_0\epsilon \nvdash_{E_{dy+}} K_{AB}$). By Definition 3.4.2 (i), we have $\langle E_{dy}, T, \sigma_0 \rangle \not\models \mathbf{K}\texttt{re}(x\sigma_0)$.

Similar to Example 8, we let σ be an arbitrary substitution satisfying that $\sigma \approx_{E_{dy+},T} \sigma_0$, $u =_s \texttt{sdec}(\{N_A\}_{K_{AB}}, x) + 1$, and $v =_s \texttt{sdec}(\{N_A + 1\}_{K_{AB}}, x)$. Since $T \vdash \{u, v\}$ and $u\sigma_0 =_{E_{dy+}} v\sigma_0$, it follows from Definition 3.3.1 that $u\sigma =_{E_{dy+}} v\sigma$. That is,

$$\texttt{sdec}(\{N_A\}_{K_{AB}}, x\sigma) + 1 =_{E_{dy+}} \texttt{sdec}(\{N_A + 1\}_{K_{AB}}, x)\sigma \tag{3}$$

It is not hard to see that Equation (3) holds only if $\sigma = [K_{AB}/x]$ (i.e., $\sigma = \sigma_0$).

Consider now, $fv(x) \subseteq fv(T)$, $x\sigma =_{E_{dy+}} x\sigma_0$ for all σ such that $\sigma \approx_{E_{dy+},T} \sigma_0$. By Definition 3.4.2 (ii), $\langle E_{dy+}, T, \sigma_0\rangle \models \mathbf{K}\mathtt{dicto}(x)$.

Finally, we see that although the attacker does not know K_{AB} (i.e., $\langle E_{dy}, T, \sigma_0\rangle \not\models \mathbf{K}\mathtt{re}(x\sigma_0)$), he or she is still able to guess the K_{AB} (i.e., $\langle E_{dy+}, T, \sigma_0\rangle \models \mathbf{K}\mathtt{dicto}(x)$). We will make this point more precise in Chapter 7.1.

3.5 Recognizability Revised

By simplifying the knowledge state from $((E, T, \Phi, \mathcal{K}), \sigma)$ to a triple $\langle E, T, \sigma\rangle$, we can handle knowledge update much easier. Suppose that an agent's knowledge state is $\vec{T} = \langle E, T, \sigma\rangle$. Let z denote a potentially ambiguous incoming message that is intended to be t. Then, after receiving the incoming message, following the discussion in Section 3.2, the agent's knowledge state is updated to

$$\vec{T} = \langle E, T \cup \{z\}, \sigma[t/z]\rangle$$

Definition 3.5.1 (Recognizability). Let $\vec{T} = \langle E, T, \sigma\rangle$ be one's knowledge state and t be a potentially ambiguous message (denoted by z). Then, we say that t is *recognizable* by $\vec{T}$ and write $\vec{T} \rhd t$, if and only if $\langle E, T \cup \{z\}, \sigma[t/z]\rangle \models \mathbf{K}\mathtt{dicto}(z)$.

Example 10. Let $\vec{T_0} = \langle E_{dy}, T_0, \epsilon\rangle$ be Alice's knowledge state, where

$$T_0 = \{K_B^+, \{N_A\}_{K_B^+}\}$$

Since $T_0\epsilon \nvdash_{E_{dy}} K_B^-$, $\vec{T_0} \not\models \mathbf{K}\mathtt{re}(K_B^-)$ follows from Definition 3.4.2 (i).

Consider a potentially ambiguous message K_B^- (denoted by z). Let σ' be an arbitrary substitution satisfying that $\sigma \approx_{E_{dy},T} \sigma$, where $T = T_0 \cup \{z\} = \{K_B^+, \{N_A\}_{K_B^+}, z\}$ and $\sigma = [K_B^-/z]$. Further, we let $u =_s \mathtt{penc}(\mathtt{pdec}(\{N_A\}_{K_B^+}, z), K_B^+)$ and $v =_s \{N_A\}_{K_B^+}$. Since $T \vdash \{u, v\}$ and $u\sigma =_{E_{dy}} v\sigma$, it follows from Definition 3.5.1 that $u\sigma' =_{E_{dy}} v\sigma'$.

That is,

$$\begin{aligned} \mathtt{penc}(\mathtt{pdec}(\{N_A\}_{K_B^+}, z), K_B^+)\sigma' &=_s \mathtt{penc}(\mathtt{pdec}(\{N_A\}_{K_B^+}, z\sigma'), K_B^+) \\ &=_{E_{dy}} \{N_A\}_{K_B^+}\sigma' \qquad (4) \\ &=_s \{N_A\}_{K_B^+} \end{aligned}$$

It is not hard to see that Equation (4) holds if and only if $\sigma' = [K_B^-/x]$ (i.e., $\sigma' = \sigma$). Consider now, $fv(z) \subseteq fv(T)$, $z\sigma' =_{E_{dy}} z\sigma$ for all σ' such that $\sigma' \approx_{E_{dy},T} \sigma$. By Definition 3.4.2 (ii), $\langle E_{dy}, T, \sigma_0\rangle \models \mathbf{K}\mathtt{dicto}(z)$. Therefore, we see from Definition 3.5.1 that K_B^- is recognizable by $\vec{T_0}$ (i.e., $\vec{T_0} \rhd K_B^-$).

Similarly, it can be shown that $\vec{T_0} \not\models \mathbf{K}N_A$ and yet N_A is recognizable (i.e., $\vec{T_0} \rhd N_A$).

In the above example, although neither K_B^- nor N_A is known (i.e., $\vec{T_0} \not\models \mathbf{K}K_B^-$ and $\vec{T_0} \not\models \mathbf{K}N_A$, Alice is still able to recognize them (i.e., $\vec{T_0} \rhd K_B^-$ and $\vec{T_0} \rhd N_A$). The following proposition makes it easier to reason about recognizability.

Proposition 3.5.2. Let $\vec{T} = \langle E, T, \sigma\rangle$ be one's knowledge state. A potentially ambiguous message t is recognizable by $\vec{T}$ if and only if $z\sigma' =_E t$ for all σ' satisfying $\sigma' \approx_{E,T\cup\{z\}} (\sigma[t/z])$ where z is a fresh variable.

CHAPTER 4: REDUCING RECOGNIZABILITY TO CONSTRAINT SOLVING

This and the next chapters address the problem of deciding recognizability under the standard intruder model. Most of the results presented in both chapters are reported in our previous paper [73].

As the notion of recognizability is based on the traditional notion of knowledge (i.e., deducibility), the problem of deciding recognizability is at least as hard as the problem of deciding deduction. Since the problem of deciding deducibility (i.e., $\vdash_E$) is undecidable in general, it is unlikely to establish general decidability results for recognizability. We thus restrict our consideration to the standard Dolev-Yao model in the hope of decidable results.

This chapter gives an overview of the main components necessary for deciding recognizability under the standard Dolev-Yao model. The next chapter explains the final construction for obtaining a decision procedure of recognizability under the Dolev-Yao model.

4.1 Ground-Explicit-Knowledge Assumption

To simplify the construction, we make another assumption other than the standard Dolev-Yao intruder model. The assumption we made here is to be used in this and the next chapters. We stress the assumption identified here is only for presentation purpose; it should not affect results presented in this and the next chapters.

As in the definition of recognizability, the agent's knowledge state is represented by a triple $\langle E, T, \sigma \rangle$. Each free variable in T represents a potentially ambiguous message and a fresh variable (e.g., z in Definition 3.5.1) represents the incoming ambiguous message. So, there could be other ambiguous messages for the agent, other than the incoming message that he or she attempts to recognize.

To simplify our presentation, we assume that the incoming message is the only ambiguous message and the expected interpretation of the incoming message is also a ground term. That is, T is a ground term set and t is a ground term, where t is the intended incoming message. Thus, $\sigma = \epsilon$ for $Dom(\sigma) \subseteq fv(T) = \emptyset$. This is called the *ground-explicit-knowledge assumption*.

With the ground-explicit-knowledge assumption, the original definition of recognizability (Definition 3.5.1) can be greatly simplified. Given an equational theory E, the knowledge state of an agent can be fully captured by a ground term set.

Proposition 4.1.1. Let E be an equational theory, T be a ground term set, and t be a ground term. Then, $\langle E, T, \epsilon \rangle \rhd t$ if and only if the following condition holds:

$$\sigma \approx_{E,T\cup\{x\}} \sigma_0 \text{ if and only if } \sigma =_E \sigma_0$$

where $\sigma_0 = [t/x]$.

We will often use $T \rhd t$ as a shorthand for $\langle E, T, \epsilon \rangle \rhd t$ when T is a ground term set and E is clear from context. Instead of working on $\vec{T} \rhd t$, in this and the next chapter we aim to give a procedure to decide $T \rhd t$.

In the following, we explain why under the standard Dolev-Yao intruder model, the problem of deciding recognizability (i.e., $\vec{T} \rhd t$) reduces to a greatly simplified problem of deciding $T \rhd t$. To understand the reason, let us consider a knowledge state $\langle E, T, \sigma\rangle$, in which the term set T that does contain variables. Then, in general

$$fv(T) = X_1 \cup X_2$$

where $X_1 = fv(T)\backslash Dom(\sigma)$ and $X_2 = fv(T) \cap Dom(\sigma)$. To validate the ground-explicit-knowledge assumption, we need to somehow get rid of variables in $fv(T)$.

For a variable x_1 in X_1, it is clear that $x_1 \notin Dom(\sigma')$ for all σ' such that $\sigma' \approx_{E,T\cup\{z\}} \sigma[t/z]$, where z is a fresh variable representing the potentially ambiguous message t. Then,

$$x_1\sigma' =_s x_1 =_s x_1\sigma[t/z]$$

for all $x_1 \in X_1$. In other words, x_1 can be replaced by a constant symbol that never occurs in T; this is analogous to the role played by Skolemization [61] in logic, where the newly generated constants are called Skolem constants [39, 97].

For a variable x_2 in X_2, since $x_2 \in Dom(\sigma)$, it does represent an ambiguous message that has certain expectation (i.e., $x_2\sigma$). As it may have different interpretations for different σ' such that $\sigma' \approx_{E,T\cup\{z\}} \sigma[t/z]$, we can not use Skolemization-like technique to eliminate this type of variables. Note that, however, every $x_2 \in X_2$ represent a new ambiguous message. If we figure out a way to use a single ambiguous message to stand for multiple ambiguous messages, then we still harbor the hope of avoiding this type of variables. The trick is, under the Dolev-Yao model, we indeed can expect a single variable (i.e., the one originally used to represent the incoming message) to account for multiple ambiguous messages. For instance, if we use $z, y_1, y_2, \cdots, y_n$ to represent ambiguous messages $t, s_1, s_2, \cdots, s_n$, respectively. Then, we may use the a new variable, say z', to represent a new ambiguous message $t' =_s t \cdot s_1 \cdot s_2 \cdots s_n$. By reasoning about the new variable z', it essentially accounts for all the original ambiguous messages $t, s_1, s_2, \cdots, s_n$.

Therefore, under the standard Dolev-Yao intruder model, we can accommodate the ground-explicit-knowledge assumption without loss of generality. Even with the ground-explicit-knowledge assumption, the problem of deciding $T \triangleright t$ is as hard as the original problem of deciding $\vec{T} \triangleright t$. Again, we retain the ground-explicit-knowledge assumption here only to make the following discussion more concise.

4.2 Characterization of Equational Theory E_{dy}

Although the definition of recognizability is general enough to capture all possible ambiguous messages, it is far from clear how to implement a decision procedure for recognizability. A major inhibitor is the infeasibility to account for all operations enabled by $\vdash$, simply because the principal can perform infinitely many operations using public function symbols (i.e., `penc`, `pdec`, `senc`, `sdec`, `fst`, and `snd`). Nonetheless, we notice that not all operations are relevant for verifying a message. In our approach, we strive to identify all such "interesting operations" that are directly or indirectly relevant to finding potentially ambiguous messages.

To capture those "interesting operations", a more precise characterization of Dolev-Yao model is desired: we define $\mathcal{F}^{\diamond} = \{f f(l) | l = r \in E\}$ and say f is an *irregular* function symbol if $f \in \mathcal{F}^{\diamond}$. A term t is *regular* (or *semi-regular*) if t (or each strict subterm of t) contains no irregular function symbols. Similarly, a substitution σ is *regular* if $Ran(\sigma)$ contains no irregular function symbols. A term t is *semi-R_E-normal* if each strict subterm of t is R_E-normal. Clearly, if t is a regular term, then it is also an R_E-normal term. Likewise, if σ is a regular substitution, then it is also an R_E-normal substitution. The notion of regularity gives an easier way to determine R_E-normality, thanks to the following lemma.

Lemma 4.2.1. Given an equational theory E, we have:

(i). Every regular term is R_E-normal;

(ii). If t is regular and σ is R_E-normal, then $t\sigma$ is R_E-normal.

The following definition sets the stage for our study of recognizability under Dolev-Yao adversaries.

Definition 4.2.2 (Regular Subterm Equational Theory). Let E be an equational theory. Then, E is a *regular subterm equational theory* if and only if for every equation $l = r \in E$ the following conditions hold

- $r \subset l$ for every equation $l = r \in E$, and
- all terms in $sub(l) \cup \{r\}$ are regular.

Claim 4.2.3. E_{dy} is a convergent regular subterm equational theory.

4.3 Constraints and Reductions

Our strategy of deciding recognizability is essentially a constraint solving procedure: Step 1 (operational equivalence) incorporates "constraints" imposed by the intended message. A new substitution is obtained in Step 2 (recognizability) by solving those "constraints". Intuitively, "constraint" is the condition imposed on terms such that possible substitutions would be more restricted and thus a less general substitution is obtained. For example $\mathtt{fst}(x\sigma) \rightarrow_{R_{E_{dy}}} N_A$ is a "constraint", which holds only if $[N_A \cdot y/x] \preceq \sigma$.

This is reminiscent of the constraint-solving approach, first proposed by Millen and Shmatikov [92], used in security protocols analysis, in which verifying security properties can be reduced to solving symbolic constraints [32, 31, 93, 43].

In further text we use “constraint” (with quotes) to informally mean a common sense fact of being restricted, and constraint (without quotes) to mean either a type-I constraint or a type-II constraint, as in Definition 4.3.2.

The two key ingredients of our approach are the notions of constraint and reduction, which, as we will see, allow us to consider only a rather reduced term space. To formalize this, we will need the following definition.

Definition 4.3.1 (Markup Term Set). A *markup term set*, notated as $\overline{T}$, is a triple $\langle T, \eta, \sigma \rangle$, where σ is a ground substitution and $Dom(\eta) \subseteq fv(T)$. Given an equational theory E, we call a markup term set $\langle T, \eta, \sigma \rangle$ *well-formed* if it obeys the following conditions

- all terms in T are regular;
- $T\eta\sigma$ is a ground term set;
- both σ and η are regular.

Intuitively, σ is the expected substitution and η represents the partially solved variables. In well-designed protocols, messages should be natural. For example, protocol participants would not expect a messages like $\texttt{penc}(A, \texttt{pdec}(B, C))$. The well-formed markup term sets precisely capture this fact.

4.3.1 Constraints

Definition 4.3.2 (Constraint). Let $\overline{T} = \langle T, \eta, \sigma \rangle$ be a markup term set. Suppose that $T\eta \vdash \{u, v\}$. Then,

- (u, v) is a *type-I constraint* of $\overline{T}$, if both u and v are regular, $u \in T\eta$, $u\sigma =_s v\sigma$, and $u \neq_s v$;
- (u, v) is a *type-II constraint* of $\overline{T}$, if u is R_E-normal and semi-regular, v is regular, $v \notin \mathcal{X}$, $u \neq_E v$, and $u\sigma \rightarrow_{R_E} v\sigma$.

Claim 4.3.3. Let $\overline{T} = \langle T, \eta, \sigma \rangle$ be a markup term set. Suppose that E is a regular subterm equational theory.

(i). If (u, v) is a *type-I constraint* of $\overline{T}$ and $\mu = mgu(u, v)$, then μ is regular;

(ii). If (u, v) is a *type-II constraint* of $\overline{T}$ and μ is the most general substitution satisfying $u\mu \rightarrow_{R_E} v\mu$, then $ff(u) \in \mathcal{F}^{\diamond}$ and μ is regular.

Lemma 4.3.4. Let $\overline{T} = \langle T, \eta, \sigma \rangle$ and $\overline{T'} = \langle T, \eta, \sigma' \rangle$ be two markup term sets. Suppose that E is a convergent regular subterm equational theory, $\overline{T}$ is well-formed, and $\sigma \approx_{E,T\eta} \sigma'$. If (u, v) is a type-I (or II) constraint of $\overline{T}$, then (u, v) is also a type-I (or II) constraint of $\overline{T'}$.

Proof. There are two cases.

(*Case 1*): (u, v) is a type-I constraint. Using the definition of type-I-constraint, we observe that both u and v are regular, $u \in T\eta$, $T\eta \vdash v$, $mgu(u, v) \neq \phi$, and $u\sigma =_s v\sigma$.

Note that $\sigma \approx_{E,T\eta} \sigma'$, and $T\eta \vdash \{u, v\}$, and $u\sigma =_E v\sigma$. Using the definition of operational equivalence, we have $u\sigma' =_E v\sigma'$. Moreover, since σ' is an R_E-normal substitution and both u and v are regular terms, we see that both $u\sigma'$ and $v\sigma'$ are R_E-normal by Lemma 4.2.1 (ii). Finally, $u\sigma' =_s v\sigma'$ due to the convergence of $\rightarrow_{R_E}$ and thus (u, v) is also a type-I constraint of $\overline{T'}$.

(*Case 2*): (u, v) is a type-II constraint. Using the definition of type-I-constraint, we observe that $T\eta \vdash \{u, v\}$, u is R_E-normal and semi-regular, v is regular, $v \notin \mathcal{X}$,

$u \neq_E v$, and $u\sigma \to_{R_E} v\sigma$. It can easily be shown that $ff(u) \in \mathcal{F}^\diamond$ (see also Claim 4.3.3 (ii)).

First, we show $u\sigma' \to^{!(n)}_{R_E} v\sigma'$. Since $\sigma \approx_{E,T\eta} \sigma'$, $T\eta \vdash \{u, v\}$, and $u\sigma =_E v\sigma$, we get $u\sigma' =_E v\sigma'$ by the definition of operational equivalence. Moreover, since v is regular and σ' is R_E-normal, it follows immediately from Lemma 4.2.1 (ii) that $v\sigma'$ is R_E-normal. Note that $u\sigma' =_E v\sigma'$ and $\to_{R_E}$ is convergent. So, $u\sigma' \to^{!(n)}_{R_E} v\sigma'$.

Then, we suppose that $n = 0$. There are two cases:

(*Case 2.1*): $\|v\| = 1$. Since $v \notin \mathcal{X}$, $\|u\sigma'\| = \|v\sigma'\| = \|v\| = 1$. Hence, $\|u\| \leq \|u\sigma'\| = 1$, giving a contradiction to the fact that $ff(u) \in \mathcal{F}^\diamond$.

(*Case 2.2*): $\|v\| > 1$. Then, $ff(v) = ff(v\sigma') = ff(u\sigma') = ff(u) \in \mathcal{F}^\diamond$. This contradicts the fact that v is regular.

So, $n > 0$. Without loss of generality, we assume that $u\sigma' =_s l'\theta' \to_{R_E} r'\theta'$ for some $l' \to r' \in R_E$ and substitution θ'. Note that $u\sigma'$ is semi-R_E-normal and $r'\theta' \subset u\sigma'$. So, $r'\theta'$ is an R_E-normal term. Consider now, both $r'\theta'$ and $v\sigma'$ are R_E-normal and $r'\theta' =_E v\sigma'$. We get $v\sigma' =_s r'\theta'$, due to the convergence of $\to_{R_E}$. Therefore $u\sigma' \to_{R_E} v\sigma'$ and it is now clear that (u, v) is a type-II constraint of $\overline{T'}$. This completes the proof. $\square$

A noticeable consequence of Lemma 4.3.4 is that, constraint property does not change with respect to operational equivalent substitutions and thus a new solver could be obtained, whenever one finds a constraint. More precisely, suppose that (u, v) is a type-I (or type-II) constraint of $\langle T, \eta_1, \sigma_1\rangle$ and $\mu = mgu(u, v)$ (or the most general substitution satisfying $u\mu \to_{R_E} v\mu$). Then, it can be shown that $\mu \lessdot \sigma_1'$ for any $\sigma_1' \approx_{E,T\eta} \sigma_1$. We therefore make the following definition.

Definition 4.3.5 (Update). Let $\overline{T} = \langle T, \eta_1, \sigma_1 \rangle$. Suppose that μ is a substitution such that $Dom(\mu) \subseteq fv(T\eta_1)$. An *update* of $\overline{T}$ by μ, denoted by $\overline{T} \downarrow_\mu$, is a markup term set $\langle T, \eta_2, \sigma_2 \rangle$ such that $\eta_2 = \eta_1\mu$, $\mu\sigma_2 = \sigma_1$, and $Dom(\sigma_2) = fv(T\eta_2)$.

4.3.2 Reductions

Definition 4.3.6 (Reduction). Let $\overline{T} = \langle T, \eta, \sigma \rangle$ be a markup term set. Suppose that $T\eta \vdash u$. Then,

- (u, v) is a *type-I reduction* of $\overline{T}$, if $T\eta \vdash u$, $T\eta \nvdash v$, and $u =_s l\theta$ and $v =_s r\theta$ for some $l \to r \in R_E$ and substitution θ;
- (u, v) is a *type-II reduction* of $\overline{T}$, if u is R_E-normal, $T\eta \vdash u$, $T\eta\sigma \nvdash v$, and $u\sigma =_s l\theta$ and $v =_s r\theta$ for some $l \to r \in R_E$ and substitution θ.

The reason why we distinguish between constraint and reduction is that a constraint imposes immediate restriction on valid substitutions, whereas a reduction does not. Further, we show that the type-II reduction counterpart of Lemma 4.3.4 does not generally hold and thus no immediate restriction can be obtained. To show this, we let $T = \{\{N_A\}_{K_B^+}, x\}$, $\sigma = [K_B^-/x]$, $\sigma' = [N_B/x]$, and $u =_s \texttt{pdec}(\{N_A\}_{K_B^+}, x)$. Then, it can be shown that $\sigma \approx_{E_{dy},T} \sigma'$ and, further, (u, N_A) is a type-II-reduction of $\langle T, \phi, \sigma \rangle$. However, (u, N_A) is not a type-II-reduction of $\langle T, \phi, \sigma' \rangle$, because

$$\texttt{pdec}(\{N_A\}_{K_B^+}, x)\sigma' =_s \texttt{pdec}(\{N_A\}_{K_B^+}, N_B) \not\to_{R_{E_{dy}}} N_A$$

Claim 4.3.7. Let $\overline{T} = \langle T, \eta, \sigma \rangle$ be a markup term set. Suppose that E is a regular subterm equational theory.

(i). If (u, v) is a *type-I reduction* of $\overline{T}$, then v is regular;

(ii). If (u, v) is a *type-I reduction* of $\overline{T}$ and $T\eta\sigma$ is a ground term set, then $v\sigma$ is ground;

(iii). If (u, w) is a *type-II reduction* of $\overline{T}$, then w is R_E-normal;

(iv). If (u, w) is a *type-II reduction* of $\overline{T}$ and $T\eta\sigma$ is a ground term set, then w is ground.

Although reductions do not give any direct impact on possible substitutions, they may introduce new constraints afterwards, thanks to the transformation lemma.

Lemma 4.3.8 (Transformation Lemma). Let E be an equational theory, T be a term set, and σ_1 and σ_2 be two ground substitutions.

(i). Suppose that $T \vdash_E t$. Then, $\sigma_1 \approx_{E,T} \sigma_2$ if and only if $\sigma_1 \approx_{E,T\cup\{t\}} \sigma_2$;

(ii). Suppose that $T \vdash s$ and x never occurs in T. Let $s\sigma_1 \to^!_{R_E} w_1$ and $s\sigma_2 \to^!_{R_E} w_2$. Then, $\sigma_1 \approx_{E,T} \sigma_2$ if and only if $\sigma_1' \approx_{E,T\cup\{x\}} \sigma_2'$, where $\sigma_1' = \sigma_1 \cup [w_1/x]$ and $\sigma_2' = \sigma_2 \cup [w_2/x]$.

Proof. (i). The "If" part is trivial. We now prove the "only if" part. To prove $\sigma_1 \approx_{E,T\cup\{t\}} \sigma_2$, it suffices to show that for all terms u and v such that $T\cup\{t\} \vdash \{u, v\}$ we have $u\sigma_1 =_E v\sigma_1 \Leftrightarrow u\sigma_2 =_E v\sigma_2$. Due to the symmetry of σ_1 and σ_2, we only need to prove one direction and proof of the reverse direction can be easily obtained by a similar analysis.

Since $T \vdash_E t$, it is obvious that $T \cup \{t\} \equiv_E T$. Note that $T \vdash \{t\} \vdash \{u, v\}$. By the definition of $\vdash_E$, there exists two terms u' and v' such that $T \vdash \{u', v'\}$, $u' =_E u$, and $v' =_E v$. Clearly, $u\sigma_1 =_E u'\sigma_1$ and $v\sigma_1 =_E v'\sigma_1$. So, $u\sigma_1 =_E v\sigma_1$ implies $u'\sigma_1 =_E v'\sigma_1$. Note that $T \vdash \{u', v'\}$ and $\sigma_1 \approx_{E,T} \sigma_2$. By the definition of operational equivalence, we have $u'\sigma_2 =_E v'\sigma_2$ and thus $u\sigma_2 =_E v\sigma_2$. Likewise, it can be shown that $u\sigma_2 =_E v\sigma_2 \Leftrightarrow u\sigma_1 =_E v\sigma_1$. Hence, $\sigma_1 \approx_{E,T\cup\{t\}} \sigma_2$.

(ii). ("If " part) To prove $\sigma_1 \approx_{E,T} \sigma_2$, it suffices to show that for all terms u and v such that $T \vdash \{u, v\}$ we have $u\sigma_1 =_E v\sigma_1 \Leftrightarrow u\sigma_2 =_E v\sigma_2$. Clearly, $T \cup \{x\} \vdash \{u, v\}$. Since $\sigma_1' \approx_{E,T\cup\{x\}} \sigma_2'$ by assumption, we have $u\sigma_1' =_E v\sigma_1' \Leftrightarrow u\sigma_2' =_E v\sigma_2'$. Note that $T \vdash \{u, v\}$ and x does not occur in T. Obviously, $x \notin fv(u)$ and $x \notin fv(v)$. So, $u\sigma_1' =_s u\sigma_1$, $v\sigma_1' =_s v\sigma_1$, $u\sigma_2' =_s u\sigma_2$, and $v\sigma_2' =_s v\sigma_2$. Therefore, $u\sigma_1 =_E v\sigma_1 \Leftrightarrow u\sigma_2 =_E v\sigma_2$.

("Only if" part) To prove $\sigma_1' \approx_{E,T\cup\{x\}} \sigma_2'$, it suffices to show that for all terms u and v such that $T \cup \{x\} \vdash \{u, v\}$ we have $u\sigma_1' =_E v\sigma_1' \Leftrightarrow u\sigma_2' =_E v\sigma_2'$.

Let $u' =_s u[x \mapsto t]$ and $v' =_s v[x \mapsto t]$. Since x never occurs in T and $T \vdash t$ by assumption, we have $T \vdash \{u', v'\}$. Note that $\sigma_1' = \sigma_1 \cup [w_1/x]$ and $\sigma_2' = \sigma_2 \cup [w_2/x]$. It is not hard to see that $u'\sigma_1 =_s u\sigma_1[x \mapsto t\sigma_1] =_E u\sigma_1[x \mapsto w_1] =_s u\sigma_1'$. So, $u\sigma_1' =_E u'\sigma_1$. Similarly, we have $v\sigma_1' =_E v'\sigma_1$, $u\sigma_2' =_E u'\sigma_2$, and $v\sigma_2' =_E v'\sigma_2$. On the other hand, since $\sigma_1 \approx_{E,T} \sigma_2$ and $T \vdash \{u', v'\}$, by the definition of operational equivalence we get $u'\sigma_1 =_E v'\sigma_1 \Leftrightarrow u'\sigma_2 =_E v'\sigma_2$. That is, $u\sigma_1' =_E v\sigma_1' \Leftrightarrow u\sigma_2' =_E v\sigma_2'$. This completes the proof. □

As the above lemma suggests, if (u, v) is a type-I reduction of $\langle T, \eta, \sigma\rangle$, then $\sigma_1 \approx_{E,T\eta} \sigma_2$ if and only if $\sigma_1 \approx_{E,T\eta\cup\{v\}} \sigma_2$. So, we can change $\langle T, \eta, \sigma\rangle$ to $\langle T \cup \{v'\}, \eta, \sigma\rangle$ where $v'\eta =_s v$, without losing or adding any condition(s) for operational equivalence; this is analogous to the transformation made by update in the previous section.

CHAPTER 5: DECISION PROCEDURE

This chapter explains the final construction for obtaining a decision procedure of recognizability under the Dolev-Yao model.

5.1 Our Construction

The last missing building block is the following definition, which formalizes our discussion in Chapter 4 about how a constraint or a reduction enables a useful transformation.

Definition 5.1.1 (Markup Term Set Rewriting). Let $\overline{T} = \langle T, \eta, \sigma \rangle$ be a markup term set. We define a binary relation $\rightarrow_E$ on markup term sets as follows:

- If (u, v) is a type-I constraint of $\overline{T}$, then $\overline{T} \rightarrow_E \overline{T}\downarrow_\mu$, where $\mu = mgu(u, v)$;
- If (u, v) is a type-II constraint of $\overline{T}$, then $\overline{T} \rightarrow_E \overline{T}\downarrow_\mu$, where μ is the most general substitution satisfying $u\mu \rightarrow_{R_E} v\mu$;
- If (u, v) is a type-I reduction of $\overline{T}$, then $\overline{T} \rightarrow_E \langle T \cup v', \eta, \sigma \rangle$, where v' a term satisfying $v'\eta =_s v$[3].
- If (u, v) is a type-II reduction of $\overline{T}$, then $\overline{T} \rightarrow_E \langle T \cup \{z\}, \eta, \sigma \cup [v/z] \rangle$, where z is a fresh variable.

The first feature of markup term set rewriting we obtain is that well-formedness property of markup term sets is invariant under transformations in both forward and

[3]This can be done by replacing every $x\eta \subseteq v$ with x.

backward directions.

Lemma 5.1.2 (Well-formedness Preserving). Let E be a regular subterm equational theory. Suppose that $\overline{T_0} \rightarrow_E^{*(n)} \overline{T_n}$. Then, $\overline{T_0}$ is well-formed if and only if $\overline{T_n}$ is well-formed.

Another feature of this transformation is its naturality in the sense that such a transformation will not impose or relax any restrictions on operational equivalence. The following theorem states this formally.

Lemma 5.1.3 (Naturality). Let E be a convergent regular subterm equational theory and $\overline{T} = \langle T, \phi, \sigma \rangle$ be a well-formed markup term set. Suppose that $\overline{T} \rightarrow_E^{*(n)} \overline{T_n} = \langle T_n, \eta_n, \sigma_n \rangle$. Then, $\sigma \approx_{E,T} \sigma'$ if and only if $\sigma' = [\eta_n \sigma'_n]_{Dom(\sigma)}$ for some σ'_n such that $\sigma'_n \approx_{E,T_n\eta_n} \sigma_n$.

Proof. ("If" part) We make induction on n. For the base case, $n = 0$, $\eta_0 = \phi$, and $\sigma'_0 = \sigma'$. Clearly, $\sigma' = [\phi\sigma'_0]_{Dom(\sigma)} = \sigma'_0$. Now, we suppose the claim is true for all $n \leq k$.

Induction step: $n = k + 1$. That is,

$$\begin{aligned} \overline{T} = \langle T, \phi, \sigma \rangle \rightarrow_E \overline{T_1} \rightarrow_E \cdots \\ \rightarrow_E \overline{T_k} = \langle T_k, \eta_k, \sigma_k \rangle \\ \rightarrow_E \overline{T_{k+1}} = \langle T_{k+1}, \eta_{k+1}, \sigma_{k+1} \rangle \end{aligned} \tag{5}$$

It follows from Lemma 5.1.2 that $\overline{T_k}$ is well-formed. Using the definition of well-formed markup term set, we have

- all terms in T_k are regular;
- $T_k\eta_k\sigma_k$ is a ground term set;
- both η_k and σ_k are regular.

For $\overline{T_k} \rightarrow_E \overline{T_{k+1}}$, by Definition 5.1.1, there exists a (u, v) (or (u, w)) that is either a constraint or a reduction of $\overline{T_k}$. Four cases are possible.

(*Case 1*): (u, v) is a type-I-constraint of $\overline{T_k}$. Using the definition of type-I-constraint, we observe that both u and v are regular, $u \in T_k\eta_k$, $T_k\eta_k \vdash v$, $mgu(u, v) \neq \phi$, and $u\sigma_k =_s v\sigma_k$. Let $\mu = mgu(u, v)$. By Claim 4.3.3 (i) we see that μ is regular. Moreover, by Definition 5.1.1 and Definition 4.3.5, we know that $T_{k+1} = T_k$, $\eta_{k+1} = \eta_k\mu$, and $\mu\sigma_{k+1} = \sigma_k$.

By assumption, $\sigma' = [\eta_{k+1}\sigma'_{k+1}]_{Dom(\sigma)}$ for some σ'_{k+1} such that $\sigma'_{k+1} \approx_{E,T_{k+1}\eta_{k+1}} \sigma_{k+1}$. That is, $\sigma'_{k+1} \approx_{E,T_k\eta_k\mu} \sigma_{k+1}$. It follows from Lemma 3.3.2 (ii) that

$$\mu\sigma'_{k+1} \approx_{E,T_k\eta_k} \mu\sigma_{k+1} = \sigma_k \tag{6}$$

Consider now, $\sigma' = [\eta_{k+1}\sigma'_{k+1}]_{Dom(\sigma)} = [\eta_k\mu\sigma'_{k+1}]_{Dom(\sigma)}$ and $\mu\sigma'_{k+1} \approx_{E,T_k\eta_k} \mu\sigma_{k+1} = \sigma_k$. By induction hypothesis, we get $\sigma \approx_{E,T} \sigma'$.

(*Case 2*): (u, v) is a type-II-constraint of $\overline{T_k}$. Let μ be the most general substitution satisfying $u\mu \rightarrow_{R_E} v\mu$. By Claim 4.3.3 (ii), μ is regular. Moreover, by Definition 5.1.1 and Definition 4.3.5, we know that $T_{k+1} = T_k$, $\eta_{k+1} = \eta_k\mu$, $\mu\sigma_{k+1} = \sigma_k$, and $Dom(\sigma_{k+1}) = fv(T_k\eta_{k+1})$.

By assumption, $\sigma' = [\eta_{k+1}\sigma'_{k+1}]_{Dom(\sigma)}$ for some σ'_{k+1} such that $\sigma'_{k+1} \approx_{E,T_{k+1}\eta_{k+1}} \sigma_{k+1}$. That is, $\sigma'_{k+1} \approx_{E,T_k\eta_k\mu} \sigma_{k+1}$. It follows from Lemma 3.3.2 (ii) that

$$\mu\sigma'_{k+1} \approx_{E,T_k\eta_k} \mu\sigma_{k+1} = \sigma_k \tag{7}$$

Consider now, $\sigma' = [\eta_{k+1}\sigma'_{k+1}]_{Dom(\sigma)} = [\eta_k\mu\sigma'_{k+1}]_{Dom(\sigma)}$ and $\mu\sigma'_{k+1} \approx_{E,T_k\eta_k} \mu\sigma_{k+1} = \sigma_k$. By induction hypothesis, we get $\sigma \approx_{E,T} \sigma'$.

(*Case 3*): (u, v) is a type-I-reduction of $\overline{T_k}$. By Definition 5.1.1, we have $T_{k+1} = T_k \cup \{v'\}$, $\sigma_{k+1} = \sigma_k$, and $\eta_{k+1} = \eta_k$, where v' is a term satisfying $v'\eta_k =_s v$. From the definition of type-I-reduction it is obvious that $T_k\eta_k \vdash v$.

By assumption, $\sigma' = [\eta_{k+1}\sigma'_{k+1}]_{Dom(\sigma)}$ for some σ'_{k+1} such that $\sigma'_{k+1} \approx_{E,T_{k+1}\eta_{k+1}} \sigma_{k+1} = \sigma_k$. That is, $\sigma'_{k+1} \approx_{E,T_k\eta_k\cup\{v\}} \sigma_k$. Note that $T_k\eta_k \vdash v$. It follows from Lemma 4.3.8 (i) that

$$\sigma'_{k+1} \approx_{E,T_k\eta_k} \sigma_k \tag{8}$$

Consider now, $\sigma' = [\eta_{k+1}\sigma'_{k+1}]_{Dom(\sigma)} = [\eta_k\sigma'_{k+1}]_{Dom(\sigma)}$ and $\sigma'_{k+1} \approx_{E,T_k\eta_k} \sigma_k$. By induction hypothesis, we get $\sigma \approx_{E,T} \sigma'$.

(*Case 4*): (u, w) is a type-II-reduction of $\overline{T_k}$. By Definition 5.1.1, we have $T_{k+1} = T_k \cup \{x\}$, $\sigma_{k+1} = \sigma_k \cup [w/x]$, and $\eta_{k+1} = \eta_k$, where x is a new variable that never occurs in T_k or $Ran(\eta_k)$. It can easily be shown that x does not occur in $T_k\eta_k$. By assumption, $\sigma' = [\eta_{k+1}\sigma'_{k+1}]_{Dom(\sigma)}$ for some σ'_{k+1} such that $\sigma'_{k+1} \approx_{E,T_{k+1}\eta_{k+1}} \sigma_{k+1} = \sigma_k$. That is, $\sigma'_{k+1} \approx_{E,T_k\eta_k\cup\{x\}} \sigma_k \cup [w/x]$.

At first, we see, from the definition of operational equivalence, that $Dom(\sigma'_{k+1}) = Dom(\sigma_k) \cup \{x\}$ and $fv(T_k\eta_k \cup \{x\}) \subseteq Dom(\sigma'_{k+1})$. So we can let $\sigma'_{k+1} = \theta \cup [w'/x]$ for some substitution θ satisfying $Dom(\theta) = Dom(\sigma)$ and a term w'. So,

$$\begin{aligned}
\sigma' &= [\eta_{k+1}\sigma'_{k+1}]_{Dom(\sigma)} \\
&= [\eta_k\sigma'_{k+1}]_{Dom(\sigma)} \\
&= [\eta_k\theta \cup [w'/x]]_{Dom(\sigma)} \\
&= [\eta_k\theta]_{Dom(\sigma)}
\end{aligned} \tag{9}$$

Then, we show that $\theta \approx_{T_k\eta_k} \sigma_k$. Since $\sigma'_{k+1} \approx_{E,T_k\eta_k\cup\{x\}} \sigma_k \cup [w/x]$, for any u', v' such that $T_k\eta_k \cup \{x\} \vdash \{u', v'\}$ we have $u'\sigma'_{k+1} =_E v'\sigma'_{k+1}$ if and only if $u'\sigma_{k+1} =_E v'\sigma_{k+1}$. Further, if x does not occur in u' or v', then $T_k\eta_k \vdash \{u', v'\}$, $u'\sigma'_{k+1} =_s u'\theta$,

$v'\sigma'_{k+1} =_s v'\theta$, $u'\sigma_{k+1} =_s u'\sigma_k$, and $v'\sigma_{k+1} =_s v'\sigma_k$. In other words, for any u', v' such that $T_k\eta_k \vdash \{u', v'\}$ we have $u'\theta =_E v'\theta$ if and only if $u'\sigma_k =_E v'\sigma_k$. Note that $fv(T_k\eta_k) \subseteq Dom(\theta) = Dom(\sigma_k)$. As a result, $\theta \approx_{T_k\eta_k} \sigma_k$.

Consider now, $\sigma' = [\eta_k\theta]_{Dom(\sigma)}$ and $\theta \approx_{E,T_k\eta_k} \sigma_k$. By induction hypothesis, we get $\sigma \approx_{E,T} \sigma'$.

("Only if" part) We make induction on n. For the base case, $n = 0$, $\eta_0 = \phi$, and $\sigma'_0 = \sigma'$. Clearly, $\sigma' = \sigma'_0 = [\phi\sigma'_0]_{Dom(\sigma)}$. Now, we suppose the claim is true for all $n \leq k$.

Induction step: $n = k + 1$. That is,

$$\begin{aligned} \overline{T} = \langle T, \phi, \sigma\rangle &\rightarrow_E \overline{T_1} \rightarrow_E \cdots \\ &\rightarrow_E \overline{T_k} = \langle T_k, \eta_k, \sigma_k\rangle \\ &\rightarrow_E \overline{T_{k+1}} = \langle T_{k+1}, \eta_{k+1}, \sigma_{k+1}\rangle \end{aligned} \tag{10}$$

Let $\sigma'_k \approx_{E,T_k\eta_k} \sigma_k$. By induction hypothesis, $\sigma' = [\eta_k\sigma'_k]_{Dom(\sigma)}$. Moreover, it follows from Lemma 5.1.2 that $\overline{T_k}$ is well-formed. Using the definition of well-formed markup term set, we have

- all terms in T_k are regular;
- $T_k\eta_k\sigma_k$ is a ground term set;
- both η_k and σ_k are regular.

For $\overline{T_k} \rightarrow_E \overline{T_{k+1}}$, by Definition 5.1.1, there exists a (u, v) (or (u, w)) that is either a constraint or a reduction of $\overline{T_k}$. Four cases are possible.

(*Case 1*): (u, v) is a type-I-constraint of $\overline{T_k}$. Using the definition of type-I-constraint, we observe that both u and v are regular, $u \in T_k\eta_k$, $T_k\eta_k \vdash v$, $mgu(u, v) \neq \phi$, and

$u\sigma_k =_s v\sigma_k$. Let $\mu = mgu(u, v)$. By Claim 4.3.3 (i) we see that μ is regular. Moreover, by Definition 5.1.1 and Definition 4.3.5, we know that $T_{k+1} = T_k$, $\eta_{k+1} = \eta_k\mu$, and $\mu\sigma_{k+1} = \sigma_k$.

Let $\overline{T'_k} = \langle T_k, \eta_k, \sigma'_k\rangle$, where $\sigma'_k \approx_{E,T_k\eta_k} \sigma_k$. Using Lemma 4.3.4 we see that (u, v) is a type-I-constraint of $\overline{T'_k}$ as well and thus $\overline{T'_k} = \langle T_k, \eta_k, \sigma'_k\rangle \rightarrow_E \overline{T'_{k+1}} = \langle T_{k+1}, \eta'_{k+1}, \sigma'_{k+1}\rangle$, where $T_{k+1} = T_k$, $\eta'_{k+1} = \eta_k\mu = \eta_{k+1}$, $\mu\sigma'_{k+1} = \sigma'_k$, and $Dom(\sigma'_{k+1}) = fv(T_k\eta_{k+1})$. Note that $\sigma_k \approx_{E,T_k\eta_k} \sigma'_k$. That is, $\mu\sigma_{k+1} \approx_{E,T_k\eta_k} \mu\sigma'_{k+1}$. It follows from Lemma 3.3.2 (ii) that $\sigma_{k+1} \approx_{E,T_k\eta_k\mu} \sigma'_{k+1}$. Furthermore,

$$\sigma' = [\eta_k\sigma'_k]_{Dom(\sigma)} = [\eta_k\mu\sigma'_{k+1}]_{Dom(\sigma)} = [\eta_{k+1}\sigma'_{k+1}]_{Dom(\sigma)}$$

(*Case 2*): (u, v) is a type-II-constraint of $\overline{T_k}$. Let μ be the most general substitution satisfying $u\mu \rightarrow_{R_E} v\mu$. By Claim 4.3.3 (ii), μ is regular. Moreover, by Definition 5.1.1 and Definition 4.3.5, we know that $T_{k+1} = T_k$, $\eta_{k+1} = \eta_k\mu$, $\mu\sigma_{k+1} = \sigma_k$, and $Dom(\sigma_{k+1}) = fv(T_k\eta_{k+1})$.

Let $\overline{T'_k} = \langle T_k, \eta_k, \sigma'_k\rangle$, where $\sigma'_k \approx_{E,T_k\eta_k} \sigma_k$. Using Lemma 4.3.4 we see that (u, v) is a type-II-constraint of $\overline{T'_k}$ as well and thus $\overline{T'_k} = \langle T_k, \eta_k, \sigma'_k\rangle \rightarrow_E \overline{T'_{k+1}} = \langle T_{k+1}, \eta'_{k+1}, \sigma'_{k+1}\rangle$, where $T_{k+1} = T_k$, $\eta'_{k+1} = \eta_k\mu = \eta_{k+1}$, $\mu\sigma'_{k+1} = \sigma'_k$, and $Dom(\sigma'_{k+1}) = fv(T_k\eta_{k+1})$. Note that $\sigma_k \approx_{E,T_k\eta_k} \sigma'_k$. That is, $\mu\sigma_{k+1} \approx_{E,T_k\eta_k} \mu\sigma'_{k+1}$. It follows from Lemma 3.3.2 (ii) that $\sigma_{k+1} \approx_{E,T_k\eta_k\mu} \sigma'_{k+1}$. Furthermore,

$$\sigma' = [\eta_k\sigma'_k]_{Dom(\sigma)} = [\eta_k\mu\sigma'_{k+1}]_{Dom(\sigma)} = [\eta_{k+1}\sigma'_{k+1}]_{Dom(\sigma)}$$

(*Case 3*): (u, v) is a type-I-reduction of $\overline{T_k}$. By Definition 5.1.1, we have $T_{k+1} = T_k \cup \{v'\}$, $\sigma_{k+1} = \sigma_k$, and $\eta_{k+1} = \eta_k$, where v' is a term satisfying $v'\eta_k =_s v$. Let $\overline{T'_k} = \langle T_k, \eta_k, \sigma'_k\rangle$, where $\sigma'_k \approx_{E,T_k\eta_k} \sigma_k$. Note that $T_k\eta_k \vdash_E v$ by the definition of type-I-reduction and $\sigma'_k \approx_{E,T_k\eta_k} \sigma_k$. It follows from Lemma 4.3.8 (i) that

$$\sigma'_k \approx_{E,T_k\eta_k \cup \{v\}} \sigma_k \tag{11}$$

Note that (u, v) is also a type-I-reduction of $\overline{T'_k}$. Let $\overline{T'_k} = \langle T_k, \eta_k, \sigma'_k \rangle \rightarrow_E \overline{T'_{k+1}} = \langle T'_{k+1}, \eta'_{k+1}, \sigma'_{k+1} \rangle$. By Definition 5.1.1 we have $\sigma'_{k+1} = \sigma'_k$, $\eta'_{k+1} = \eta_k$, and $T'_{k+1} = T_k \cup \{v'\} = T_{k+1}$. Consider now, $T_{k+1}\eta_{k+1} = (T_k \cup \{v'\})\eta_k = T_k\eta_k \cup \{v'\eta_k\} = T_k\eta_k \cup \{v\}$, $\sigma'_{k+1} = \sigma'_k$, and $\sigma_{k+1} = \sigma_k$. As a result, Equation (11) reduces to

$$\sigma'_{k+1} \approx_{E,T_{k+1}\eta_{k+1}} \sigma_{k+1}$$

Furthermore, $\sigma' = [\eta_k \sigma'_k]_{Dom(\sigma)} = [\eta_{k+1} \sigma'_{k+1}]_{Dom(\sigma)}$.

(*Case 4*): (u, w) is a type-II-reduction of $\overline{T_k}$. By Definition 5.1.1, we have $T_{k+1} = T_k \cup \{x\}$, $\sigma_{k+1} = \sigma_k \cup [w/x]$, and $\eta_{k+1} = \eta_k$, where x is a new variable satisfying $x \notin fv(T_k) \cup Ran(\eta_k) \cup Dom(\sigma_k)$. It can easily be shown that x does not occur in $T_k\eta_k$. Let $\overline{T'_k} = \langle T_k, \eta_k, \sigma'_k \rangle$ and $\sigma'_{k+1} = \sigma'_k \cup [w'/x]$, where $\sigma'_k \approx_{E,T_k\eta_k} \sigma_k$ and $u\sigma'_k \rightarrow^!_{R_E} w'$. Moreover, by the definition of type-II-reduction, we know that $T_k\eta_k \vdash u$ and u is semi-regular.

Note that E is a convergent subterm equational theory and w is R_E-normal by Claim 4.3.7 (iv). Obviously, $u\sigma_k \rightarrow^!_{R_E} w$. Consider now, $T_k\eta_k \vdash u$, x never occurs in $T_k\eta_k$, $u\sigma_k \rightarrow^!_{R_E} w$, $u\sigma'_k \rightarrow^!_{R_E} w'$, and $\sigma_k \approx_{E,T_k\eta_k} \sigma'_k$. It follows from Lemma 4.3.8 (ii) that $\sigma_k \cup [w/x] \approx_{E,T_k\eta_k \cup \{x\}} \sigma'_k \cup [w'/x]$. That is,

$$\sigma_{k+1} \approx_{E,T_{k+1}\eta_{k+1}} \sigma'_{k+1}$$

Furthermore,

$$
\begin{aligned}
[\eta_{k+1}\sigma'_{k+1}]_{Dom(\sigma)} &= [\eta_k\sigma'_k \cup [w'/x]]_{Dom(\sigma)} \\
&= [\eta_k\sigma'_k]_{Dom(\sigma)} \\
&= [\sigma']_{Dom(\sigma)} \\
&= \sigma'
\end{aligned} \tag{12}
$$

This completes the proof. □

Not surprisingly, with the proven salient features, markup term set rewriting enables us to find recognizable terms.

Theorem 5.1.4 (Correctness). Let T be a regular and ground term set, and $\sigma = [t/x]$ be a ground substitution. If $solve(\langle T \cup \{x\}, \phi, \sigma\rangle) = \langle T_n, \eta_n, \sigma_n\rangle$ and $x\eta_n =_s t$, then $T \rhd t$.

Proof. Let σ' be an arbitrary substitution satisfying $\sigma' \approx_{E_{dy},T\cup\{x\}} \sigma$. Then, we can apply Lemma 5.1.3 and obtain that $\sigma' = [\eta_n\sigma'_n]_{Dom(\sigma)}$ for some σ'_n such that $\sigma'_n \approx_{E,T_n} \sigma_n$. Then, $x\sigma' =_s x\eta_n\sigma'_n =_s t\sigma'_n =_s t$. Moreover, since $\sigma' \approx_{E_{dy},T\cup\{x\}} \sigma$, we get $Dom(\sigma') = Dom(\sigma) = \{x\}$. Thus, $\sigma' = [t/x] = \sigma$. Now, it is not hard to see that $\sigma' \approx_{E_{dy},T\cup\{x\}} [t/x]$ if and only if $\sigma' = [t/x]$. By the definition of recognizability, we have $T \rhd t$. □

Intuitively, the markup term set rewriting is a recognizing process; every time a markup term set is rewritten, either a constraint or a reduction is found. By collecting all the constraints and reductions, we get all information needed to recognize any proven recognizable term.

5.2 Algorithm

We now present the algorithm for markup term set rewriting, that is, given a well-formed markup term set $\overline{T}$ as input, it returns a well-formed markup term set $\overline{T'}$ such that $\overline{T} \to^{!}_{E_{dy}} \overline{T}'$. Then, in light of the correctness theorem, one can decide the recognizability accordingly.

Theorem 5.2.1 (Termination). Suppose that T is a ground term set and $\sigma = [t/x]$ is a ground substitution. Then, algorithm $solve(\langle T \cup \{x\}, \phi, \sigma\rangle)$ is terminating.

Proof. Let $\overline{T_0} = \langle T \cup \{x\}, \phi, \sigma\rangle$ and $\overline{T_0} \to^{!}_{E_{dy}} \overline{T'} = \langle T', \eta', \sigma'\rangle$. Then, $\overline{T'} = solve(\overline{T_0})$. We assume, without loss of generality, that

$$\begin{aligned} \overline{T_0} &\to^{*}_{E_{dy}} \overline{T_i} = \langle T_i, \eta_i, \sigma_i\rangle \\ &\to_{E_{dy}} \overline{T_{i+1}} = \langle T_{i+1}, \eta_{i+1}, \sigma_{i+1}\rangle \to^{!}_{E_{dy}} \overline{T'} \end{aligned} \tag{13}$$

Using Definition 4.3.5 and 5.1.1, we observe that $T_i\eta_i\sigma_i \subseteq T_{i+1}\eta_{i+1}\sigma_{i+1}$. Moreover, since E_{dy} is a regular subterm equational theory, a case-by-case analysis shows that each $t \in (T_{i+1}\eta_{i+1}\sigma_{i+1})\backslash(T_i\eta_i\sigma_i)$ occurs in $T_i\eta_i\sigma_i$. Thus, one can easily see that $T \cup \{t\} \subseteq T'\eta'\sigma'$ and each $t \in (T'\eta'\sigma')\backslash(T \cup \{t\})$ occurs in $T \cup \{t\}$. Note that the number of terms occurring in term set $T \cup \{t\}$ is bounded by $\|t\| - 1 + \sum_{u \in T}(\|u\| - 1)$. Consequently, $T'\eta'\sigma'$ is a finite term set.

To avoid any confusion, we assume the markup term set $\overline{T_i}$ as the input for Algorithm 1 in the following discussion.

Let us first analyze line (1) to (14) of Algorithm 1, which cope with reductions (either type-I or II). Each reduction (either type-I or II) would produce a new term, that is $v'\eta\sigma$ or w, in $T'\eta'\sigma'$, because both $v'\eta\sigma$ and w are ground terms, which are not

Algorithm 1 *solve*($\overline{T}$)

Input: a well-formed markup term set $\overline{T} = \langle T, \eta, \sigma \rangle$
Output: an updated markup term set

/* *type-I reduction* */
if $\exists u, v.\ u \in T\eta, T\eta \nvdash v$ and $\texttt{fst}(\text{or } \texttt{snd})(u) \rightarrow_{R_{E_{dy}}} v$ **then**
  v' is obtained by replacing every $x\eta$ in v with x, where $x \in Dom(\eta)$.
  return $solve(\langle T \cup v', \eta, \sigma \rangle)$
if $\exists u, v, s.\ u \in T\eta, T\eta \nvdash v, T\eta \vdash s$ and $\texttt{pdec}(u, s) \rightarrow_{R_{E_{dy}}} v$ **then**
  v' is obtained by replacing every $x\eta$ in v with x, where $x \in Dom(\eta)$.
  return $solve(\langle T \cup v', \eta, \sigma \rangle)$

/* *type-II reduction* */
if $\exists u, w.\ u \in \mathcal{X} \cap T\eta$, $\texttt{fst}(\text{ or } \texttt{snd})(u\sigma) \rightarrow_{R_{E_{dy}}} w$
and there does not exist a term v such that $T\eta \vdash v$ and $v\sigma =_s w$ **then**
  let z be a fresh variable
  $\overline{T} \leftarrow \langle T \cup z, \eta, \sigma \cup [w/z] \rangle$
  return $solve(\overline{T})$
if $\exists u, w, s.\ u \in T\eta, T\eta \vdash s$ and $\texttt{pdec}(u, s)\sigma \rightarrow_{R_{E_{dy}}} w$
and there does not exist a term v such that $T\eta \vdash v$ and $v\sigma =_s w$ **then**
  let z be a new variable that never occurs in $T\eta$
  $\overline{T} \leftarrow \langle T \cup z, \eta, \sigma \cup [w/z] \rangle$
  return $solve(\overline{T})$

/* *type-I constraint* */
if $\exists u, v.\ u \in T\eta, T\eta \vdash v$, v is regular, $u \neq_s v, u\sigma =_s v\sigma$ **then**
  $\overline{T} \leftarrow \overline{T} \downarrow_\mu$ where $\mu = mgu(u, v)$
  return $solve(\overline{T})$

/* *type-II constraint* */
if $\exists u, v.\ u \in \mathcal{X} \cap T\eta, T\eta \vdash v$, v is regular, $v \notin \mathcal{X}$, and
$\texttt{fst}(\text{ or } \texttt{snd})(u\sigma) \rightarrow_{R_{E_{dy}}} v\sigma$ **then**
  $\overline{T} \leftarrow \overline{T} \downarrow_\mu$ where μ is the most general substitution
  satisfying $\texttt{fst}(\text{ or } \texttt{snd})(u\mu) \rightarrow_{R_{E_{dy}}} v\mu$
  return $solve(\overline{T})$
if $\exists u, v, s.\ u \in T\eta, T\eta \vdash v$, v is regular, $v \notin \mathcal{X}$, $T\eta \vdash s$, and
$\texttt{pdec}(u, s)\sigma \rightarrow_{R_{E_{dy}}} v\sigma$ **then**
  $\overline{T} \leftarrow \overline{T} \downarrow_\mu$ where μ is the most general substitution satisfying
    $\texttt{pdec}(u, s)\mu \rightarrow_{R_{E_{dy}}} v\mu$
  return $solve(\overline{T})$
return $\overline{T}$

subject to change in markup term set rewriting. As a result, the number of reductions explored by the algorithm is also bounded.

Now, we turn to line (15) to (23) of Algorithm 1, which cope with constraints (either type-I or II). It's important to note that to build a constraint, say (u, v), there must exist a term $u_0 \in T_i\eta_i$. Then, $u_0\sigma_i \in T_i\eta_i\sigma_i \subseteq T'\sigma'\eta'$. Since $u_0\sigma_i$ is ground and subject to no change, there is exactly one $u_0\sigma_i \in T'\sigma'\eta'$. Though not unique, such terms u'_0 that satisfy $u'_0\sigma'_i =_s u_0\sigma_i$ is finite, simply because $T'_i\eta'_i$ is a finite term set. The number of constraints explored by the algorithm is therefore bounded.

Finally, we conclude that $solve(\langle T \cup \{x\}, \phi, \sigma\rangle)$ is terminating. □

We do not address computational complexity here due to the fact that efficiency is not a major concern in deciding recognizability. However, we claim without proof that, the problem of deciding recognizability under standard Dolev-Yao model can be solved in polynomial time.

CHAPTER 6: TOWARDS THE ATTACKER'S VIEW OF PROTOCOL NARRATIONS (OR, COMPILING SECURITY PROTOCOLS)

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 chapter, we establish a faithful view of the attacker based on the notion of recognizability, which offers 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. Our work can be seen as part of continuing efforts in compiling security protocols, which aims at semantics for protocol narrations.

The results presented in this chapter are mainly reported in our previous papers [72, 75].

6.1 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 [96].

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 [28].

Second, the implementor's view of protocol narrations can be "pessimistic", because how principals check incoming messages is often neglected in protocol narrations [2]. 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 [25]. For example, Ceelen et al. [23] show that Lowe's modified KSL protocol [83] 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 work, 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 3. Unlike most previous work which has focused on formalization or compilation [22, 21, 19, 94], 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.

6.1.1 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 part of protocol specification [2]. 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

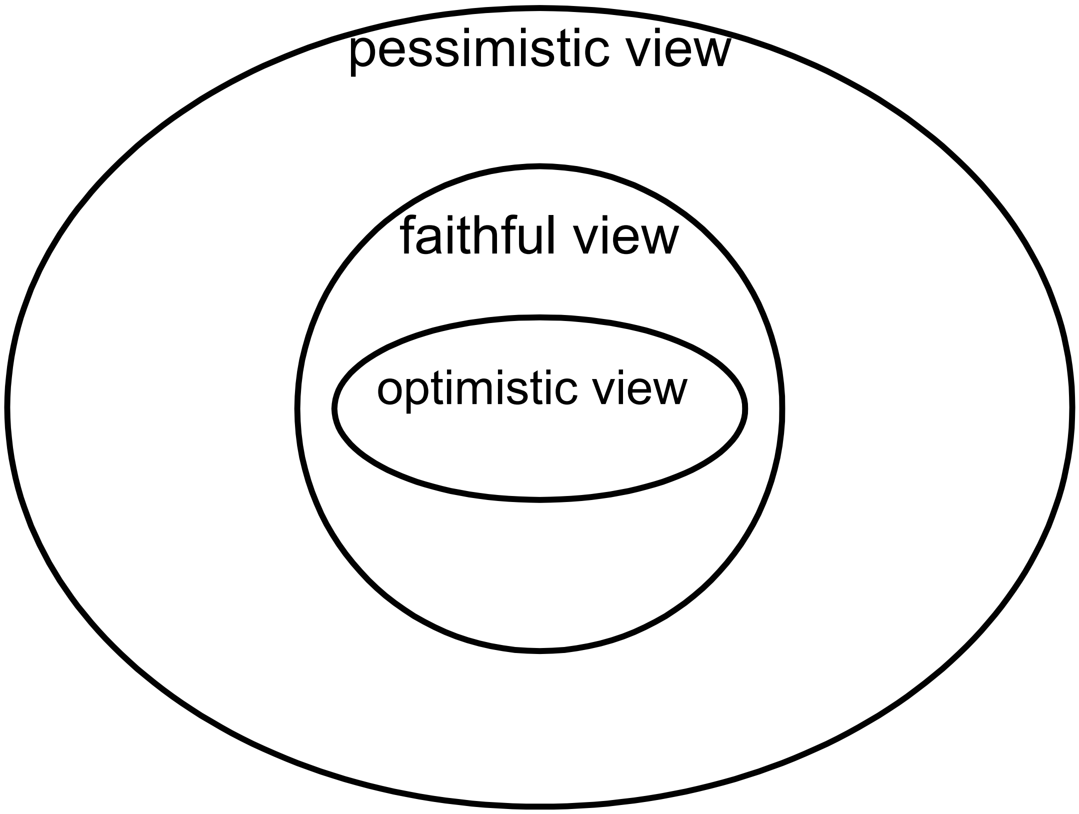


Figure 3: Sets of possible protocol execution traces under different views

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* [72] 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 [73].

We then use this ideal semantics to guide protocol implementation by deriving *all* necessary equality checks. Similar to [25], 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 $M, \{N_A, \underline{M, A, B}\}_{K_{AS}}$.

6.1.2 Contributions

The main contributions of this chapter 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 [73].
- 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 6.4.3). 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 chapter is organized as follows: Section 6.2 is dedicated to the interpretations of exchanged messages in protocol narrations. Section 6.3 gives the ideal semantics of protocol narrations based on interpretations of the exchanged messages. In light of this semantics, Section 6.4 presents our classification of protocol implementations and attacks. Section 6.6 discusses related work. Section 6.7 concludes the chapter and outlines the future work.

6.2 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 *knowledge state* 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.

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. We will continue to use knowledge states to account for uncertainty. We continue to use $\vec{T} = \langle E, T, \sigma \rangle$ to encapsulate one's epistemic state with uncertainty. In Section 3.5, we have also used operational equivalence to characterize one's inability to discriminate two interpretations of a message. Hereafter, we make explicit mention of each principal's initial knowledge before a protocol run.

Example 11. To model principals' knowledge after completion of the Otway-Rees protocol, we use T_{A0}, T_{B0}, and T_{S0} to represent the initial explicit knowledge of A, B, and C, 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

$$\vec{T_A} = \langle T_{A0} \cup \{x_4\}, \sigma_A\rangle$$

$$\vec{T_B} = \langle T_{B0} \cup \{x_1, x_3\}, \sigma_B\rangle$$

$$\vec{T_S} = \langle T_{S0} \cup \{x_2\}, \sigma_S\rangle$$

where $x_1, \cdots, x_4$ represents the four incoming ambiguous messages, and

$$\sigma_A = [\{N_A \cdot K_{AB}\}_{K_{AS}}/x_4]$$

$$\sigma_B = [(M \cdot A \cdot B \cdot \{N_A \cdot M \cdot A \cdot B\}_{K_{AS}})/x_1, \{N_B \cdot K_{AB}\}_{K_{BS}})/x_3]$$

$$\sigma_S = [(A \cdot B \cdot \{N_A \cdot M \cdot A \cdot B\}_{K_{AS}}, \{N_B \cdot M \cdot A \cdot B\}_{K_{BS}})/x_2]$$

Example 12. Consider again the Otway-Rees protocol. As in Example 11, the initial explicit knowledge of each principal is given by T_{A0}, T_{B0}, and T_{S0}, respectively. Then, $\vec{T}_{B0} = \langle T_{B0}, \epsilon\rangle$ is B's initial knowledge state. After receiving the first message, the knowledge state of B becomes $\vec{T}_{B1} = \langle T_{B1}, \sigma_1\rangle$, where $T_{B1} = T_{b0} \cup \{x, y\} = \{A, B, S, N_B, K_{bs}, x, y\}$ and $\sigma_1 = [M/x, \{N_A \cdot M \cdot A \cdot B\}_{K_{AS}}/y]$.

It can be shown that $\vec{T}_{B0} \not\triangleright \{N_A \cdot M \cdot A \cdot B\}_{K_{AS}}$. In other words, B does not recognize the message $\{N_A \cdot M \cdot A \cdot B\}_{K_{AS}}$. However, the message $\{N_A \cdot M \cdot A \cdot B\}_{K_{AS}}$ should not be simply treated as a black box to B, because otherwise y can be interpreted as an arbitrary message. To see why this is not acceptable, we let $\sigma_1' = [N_A \cdot N_A/y]$, $u =_s \mathtt{fst}(y)$, and $v =_s \mathtt{snd}(y)$. Note that $T_{B1} \vdash \{u, v\}$, $u\sigma_1' =_{E_{dy}} v\sigma_1' =_{E_{dy}} N_A$, 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 $u\sigma_1' =_{E_{dy}} v\sigma_1'$. Thus, $\sigma_1 \not\approx_{E_{dy},T_{b1}} \sigma_1'$ follows immediately from Definition 3.5.1. In other words, if y is interpreted as the message $N_A \cdot N_A$, then B would be able to distinguish it from the intended message $\{N_A \cdot M \cdot A \cdot B\}_{K_{AS}}$.

Although the notion of recognizability offers a rigorous and yet intuitive way to interpret ambiguous messages, we may not be able to apply it directly here. The original definition of recognizability (Definition 3.5.1) intends to formalize the intuitive understanding of verifying a message. The definition is amenable to the situation when a message is recognizable. For a message that is not recognizable, recognizability does not characterize to what extent the message can be recognized. Indeed, we can treat a recognizable message as a white box, but it is unreasonable to treat an unrecognizable message simply as a black box, as we have seen in in Example 12, because we may still hold some expectation of the message. We thus extend the original definition of recognizability to capture the fact to what extent a message can be understood.

Definition 6.2.1 (Solver). Let $\vec{T} = \langle E, T, \sigma_0 \rangle$ be a knowledge state and let $X = fv(T)$. We say that substitution θ is a solver for $\vec{T}$ if and only if the following conditions hold

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

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

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

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

Definition 6.2.2 (Recognized As). Let $\vec{T} = \langle E, T, \sigma_0 \rangle$ be a knowledge state and t be a ground term. We say that t is *recognized as* t' by $\vec{T}$ if and only if there exists a solver θ for $\langle E, T \cup \{x\}, \sigma_0 \circ [t/z] \rangle$ such that $z\theta =_E t'$, where z is a fresh variable.

Clearly, a term t is recognizable by $\vec{T}$ if and only if t is recognized as itself by $\vec{T}$.

Lemma 6.2.3. $\vec{T} \rhd t$ if and only if t is recognized as itself by $\vec{T}$.

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

Example 13. Let us consider the following ASW protocol, which is proposed by Asokan et. al. [8] for fair exchange and contract signing.

Message 1. $A \to B : \{K_A^+, K_B^+, M, hash(N_A)\}_{K_A^-}$

Message 2. $B \to A : \{\{K_A^+, K_B^+, M, hash(N_A)\}_{K_A^-}, hash(N_B)\}_{K_B^-}$

Message 3. $A \to B : N_A$

Message 4. $B \to A : N_B$

We assume that the initial explicit 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 state 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 $x_1, \cdots, x_4$ 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 $(T_{A0} \cup \{x_2, x_4\}) \vdash \{u_1, \cdots, u_4\}$ 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 if and only if

$$x_2\sigma_A =_{E_{dy}} \{\{K_A^+ \cdot K_B^+ \cdot M \cdot \mathtt{hash}(N_A)\}_{K_A^-} \cdot \mathtt{hash}(x_4)\sigma_A\}_{K_B^-}$$

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 solver for $\vec{T}_A$. In fact, θ_A is the only 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 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 knowledge state. To determine 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 [92]. Here, we extend the constraint solving approach used in Chapter 4 to find a MCS.

A *constraint* of a knowledge state $\langle E, T, \sigma\rangle$ 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 $\mathcal{C} \rightsquigarrow \Theta$, 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 6.2.4 (Constraint Base). Let $\vec{T} = \langle E, T, \sigma \rangle$ be a knowledge state. Suppose that $\mathcal{C}$ is the set of all constraints of $\vec{T}$ under E and $\mathcal{C} \rightsquigarrow \Theta$. 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 $\mathcal{C}' \rightsquigarrow \Theta$ and $\mathcal{C}'$ is finite.

This is analogous to the definition "finite basis property" given in [25]. In Example 13, we see $\{(u_1, u_2), (u_3, u_4)\}$ is a constraint base of $\vec{T}_A$.

Proposition 6.2.5. Let $\vec{T} = \langle T, \sigma \rangle$ be a knowledge state. Suppose that $\mathcal{C}$ is a constraint base of $\vec{T}$. Then, $\vec{T} \rightsquigarrow \Theta$ if and only if $\mathcal{C} \rightsquigarrow \Theta$.

In view of Proposition 6.2.5, 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 [100, Chapter 8]. Nonetheless, restricting ourselves to some specific equational theories is likely to yield decidable results. Notably, a procedure is given in [73] 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.

6.3 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 [52], a widely-used formalism in modeling and verifying security protocols [60, 103, 92]. 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).

6.3.1 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 attacker; 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 4 shows a bundle that illustrates the expected execution of the ASW protocol.

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 4 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,

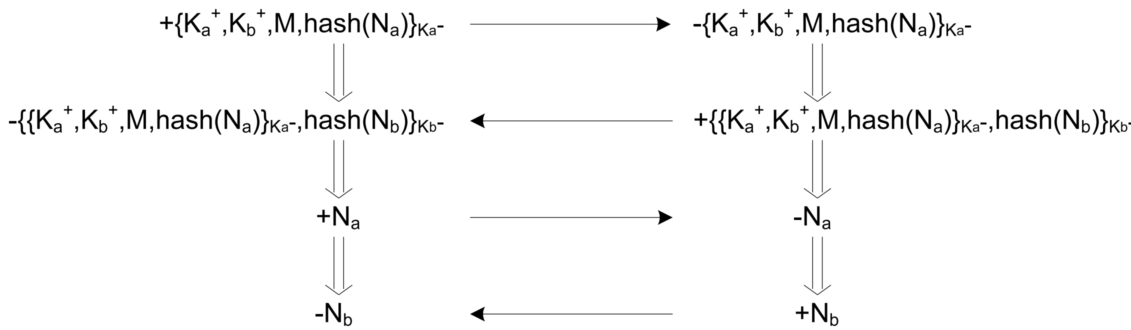


Figure 4: ASW protocol: a bundle.

role A in the ASW protocol should be specified by

$$\underline{A[M, A, B, N_A, x]}$$

$$\langle + \{K_A^+ \cdot K_B^+ \cdot M \cdot \mathtt{hash}(N_A)\}_{K_A^-},$$

$$- \{\{K_A^+ \cdot K_B^+ \cdot M \cdot \mathtt{hash}(N_A)\}_{K_A^-}, \mathtt{hash}(x)\}_{K_B^-},$$

$$+ N_A, -x\rangle$$

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

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

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

6.3.2 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* if and only if $\vec{\mathcal{K}}(\vec{s}_1) \downarrow_{ts}$ 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 $\vec{s}_1 \approx_E \vec{s}_2$, if and only if $\vec{\mathcal{K}}(\vec{s}_1) \downarrow_{subs} \approx_{E,T} \vec{\mathcal{K}}(\vec{s}_2) \downarrow_{subs}$ where $T = \vec{\mathcal{K}}(\vec{s}_1) \downarrow_{ts} = \vec{\mathcal{K}}(\vec{s}_2) \downarrow_{ts}$.

Definition 6.3.1. Given an equational theory E, we say that an execution trace *tr* is *in compliance with* a set of strands $\vec{S}$, written as $\vec{S} \rightsquigarrow tr$, if and only if $tr \approx_E \vec{s}$ 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 $\vec{S}_2$.

6.3.3 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 6.3.2 (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 if and only if $\vec{S} \approx_E TR_0$.

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 [103, 21, 40, 13].

Our definition of "recognized as" (Definition 6.2.2) 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 successful protocol run.

Altogether, we obtain Algorithm 2 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.

Algorithm 2 *ExtractIdealSemantics*

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

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 6.2.1, it is operationally equivalent to $\vec{\mathcal{K}}(tr)$ and is in its most general form. So, in line 11, we include all strands associated with those interpretations in output ideal semantics.

Theorem 6.3.3. Let TR_0 be a set of narrative traces. Then,

$ExtractIdealSemantics(TR_0, E)$ returns an ideal semantics of TR_0.

Proof. Let $\vec{S}_I = ExtractIdealSemantics(TR_0, E)$. 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 $TR_0 \rightsquigarrow tr$, 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 6.2.1, 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 2 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, $\vec{S}_I \rightsquigarrow tr$.

("Only if" part) By $\vec{S}_I \rightsquigarrow tr$, we see from Algorithm 2 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 a 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 $TR_0 \rightsquigarrow tr$. □

We stress that a protocol could be executed in a hostile environment. A principal may intentionally abort a protocol before completion. So, in Algorithm 2 the narrative traces must include all partial protocol runs [34]. To highlight the effect of partial runs on the ideal semantics, let us consider an example.

Example 14. We consider the following contrived protocol:

$$\begin{array}{ll} \text{Message 1.} & A \rightarrow B : M_1 \\ \text{Message 2.} & B \rightarrow A : M_2 \\ \text{Message 3.} & A \rightarrow B : M_3 \\ \text{Message 4.} & B \rightarrow A : M_4 \end{array}$$

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:

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

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

6.4 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 13) A ought to check whether $\texttt{fst}(\texttt{pdec}(x_2, K_B^+))$ equals to the first sent message, where x_2 signifies the received message.

To specify internal actions, we define a *check* event as $\mathit{check}(u = v)$ or $\mathit{check}(u \neq v)$, 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 if and only if 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 [21].

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 6.4.1 (In Compliance with). An execution trace tr is *in compliance with* a protocol implementation $\mathcal{P}$ if and only if 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 $\mathcal{P}_1 \approx_E \mathcal{P}_2$, if and only if $\mathcal{P}_1 \subseteq_E \mathcal{P}_2$ 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.

6.4.1 Ideal Implementation

Definition 6.4.2 (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 6.4.3. 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. ("If" part) As we will see in the next subsection, Algorithm 3 gives an implementation $\mathcal{P}$. To prove $\mathcal{P} \approx_E \vec{S}$, by Definition 6.3.2 it suffices to show that $\mathcal{P} \approx_E TR_0$. That is, $\mathcal{P} \rightsquigarrow tr \Leftrightarrow \vec{S} \rightsquigarrow tr$.

We begin with the "$\Rightarrow$" direction. By $\mathcal{P} \rightsquigarrow tr$, 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 $\mathcal{C} \rightsquigarrow \Theta$. We see from Definition 6.4.1 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 6.2.5 that $\vec{\mathcal{K}}(tr_0) \rightsquigarrow \Theta$. By Definition 6.2.1, 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 $TR_0 \rightsquigarrow tr$. 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 $\{N_A, K_A^+, x\} \vdash t$. This completes the proof. □

6.4.2 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 6.4.4 (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 knowledge state $\vec{T}$, 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:

$$
\begin{aligned}
&\underline{A[M, A, B, N_A, x_2, x_4]}\\
&\langle \{M, A, B, K_A^+, K_B^+, K_A^-, N_A\},\\
&\quad + \{K_A^+ \cdot K_B^+ \cdot M \cdot \mathtt{hash}(N_A)\}_{K_A^-},\ -x_2,\\
&\quad \mathit{check}(\mathtt{fst}(\mathtt{pdec}(x_2, K_B^+)) = \{K_A^+ \cdot K_B^+ \cdot M \cdot \mathtt{hash}(N_A)\}_{K_A^-}),\\
&\quad + N_A,\ -x_4,\ \mathit{check}(\mathtt{snd}(\mathtt{pdec}(x_2, K_B^+)) = \mathtt{hash}(x_4))\rangle
\end{aligned}
$$

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 Lemma 2.1.3, 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 15. To make this more concrete, let us consider a very simple protocol.

$$\begin{array}{ll} \text{Message 1.} & A \rightarrow B : \{K_{AB}\}_{K_B^+} \\ \text{Message 2.} & B \rightarrow A : \{M\}_{K_{AB}} \end{array}$$

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 $t'_2 =_s \texttt{penc}(M, \texttt{pdec}(x_1, K_B^-)$, 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

$$\begin{aligned}\vec{s}_1 = \langle T_{B0}, -x_1, &check(\mathtt{pdec}(x_1, K_B^-) = K_{AB}), \\ &+ \{M\}_{K_{AB}}\rangle\end{aligned} \tag{14}$$

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

$$\begin{aligned}\vec{s}_2 = \langle T_{B0}, -x_1, &check(\mathtt{pdec}(x_1, K_B^-) = K_{AB}), \\ &+ \mathtt{penc}(M, \mathtt{pdec}(x_1, K_B^-))\rangle\end{aligned} \tag{15}$$

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 6.4.5. 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 6.4.5.

Theorem 6.4.6. Let TR_0 be a set of narrative traces and $\vec{S}$ be an ideal semantics of

Algorithm 3 *DerivePrudentImplementation*

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

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

TR_0. Then, $Derive - Prudent - Implementation(TR_0)$ returns an prudent implementation of $\vec{S}$.

6.4.3 Refined Implementation

To illustrate the idea of implementation refinement, let us reexamine the motivating example given in Section 6.1. We recapitulate the well-known type-flaw attack here.

$$
\begin{array}{lcl}
1.\ A \rightarrow B & : & M, A, B, \{N_A, M, A, B\}_{K_{AS}} \\
4.\ I(B) \rightarrow A & : & M, \{N_A, M, A, B\}_{K_{AS}}
\end{array}
$$

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 attacker I. The attacker 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

$$
\begin{aligned}
& \underline{A[M, A, B, S, N_A, K_{AS}, x]} \\
& \langle \{M, A, B, S, N_A, K_{AS}\}, \\
& \quad + M \cdot A \cdot B \cdot \{N_A \cdot M \cdot A \cdot B\}_{K_{AS}}, \\
& \quad - x_4, \ \mathit{check}(\{N_A \cdot M \cdot A \cdot B\}_{K_{AS}} \neq x) \rangle
\end{aligned}
$$

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 [80] on the Needham-Schroeder public-key authentication protocol [95] can not be thwarted by adding any *check* event(s).

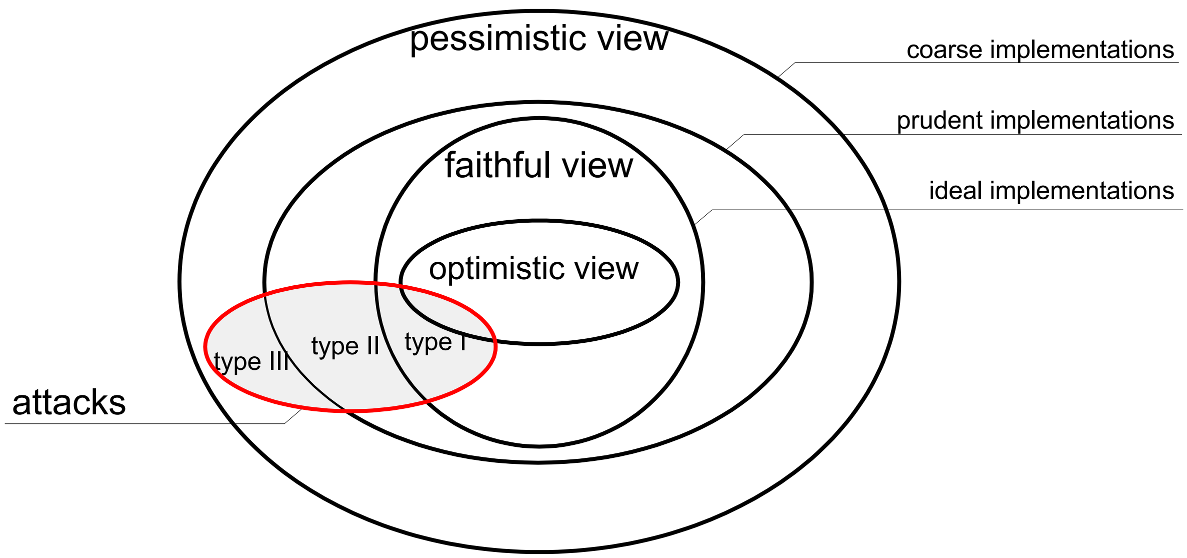
In general, a known attack can be categorized into the following three types:

- *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 5.

6.5 Application to Type-flaw Attacks

Many security protocols are vulnerable to type-flaw attacks, in which a protocol message may be subsequently forged from another message. Let us again consider the Otway-Rees protocol [96]:



Note: refined implementation = prudent implementations - type II attacks

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

Figure 5: Classification of protocol implementations and attacks

$$A \rightarrow B : M, A, B, \{N_A, M, A, B\}_{K_{AS}}$$

$$B \rightarrow S : M, A, B, \{N_A, M, A, B\}_{K_{AS}}, \{N_B, M, A, B\}_{K_{BS}}$$

$$S \rightarrow B : M, \{N_A, K_{AB}\}_{K_{AS}}, \{N_B, K_{AB}\}_{K_{BS}}$$

$$B \rightarrow A : M, \{N_A, K_{AB}\}_{K_{AS}}$$

After executing the first three messages, principal A is expecting a K_{AB}, which is a symmetric key shared between A and B, from the trusted third party S. The shared key K_{AB} is dynamically generated by S and A does not have any prior knowledge about the bit string. Therefore, any message of the form $M, \{N_A, t\}_{K_{AS}}$ would be accepted by A, as long as the bit string length of t equals to that of K_{AB}. Thus, an attacker can easily replay the message $\{N_A, M, A, B\}_{K_{AS}}$ to A and then A would use M, A, B as the secret if the length satisfies the requirement.

Various approaches have been proposed to defend against type-flaw attacks. Heather et al. [65] propose a tagging scheme to prevent type-flaw attacks, in which tags are used to label each field of a message with its intended type. However, since tag information can potentially be confused with data [87], a tagged protocol may give rise to more intricate attacks. More importantly, the question of whether an existing protocol (without any change) is vulnerable to type-flaw attack is not answered.

Catherine Meadows [89] develops a formal model of types to characterize one's capability to verify messages. Without exploring the intuitive idea behind, the procedure of verifying the locality of types could be rather complicated. In [79, 78], Z specification language is employed to model ambiguous messages. The approach based on Z specification language cannot be directly applied to existing protocol analysis tools in a straight-forward way.

However, most of existing approaches are heuristic without giving a satisfiable answer to the very first question:

Why can a security protocol be type-flawed?

Rather than developing one particular defense mechanism against type-flaw attacks, we pursue to answer this question by exploring a principal's ability/inability to cope with ambiguous messages.

In fact, a protocol could be type-flawed if a message could not be "verified" by the receiver. As we have seen in this chapter, the notion of recognizability enables us to precisely capture to what extent a message can be understood through protocol compilation.

More importantly, we notice that for most type-flaw attacks there are visible difference to the protocol participants as shown in Section 6.4.3. In other words, most type-flaw attacks are type-II attacks and thus can be prevented through implementation refinement.

6.6 Discussion and Related Work

Starting with the early work of Carlsen [22], a lot of efforts have been made to formalize security protocol descriptions or to devise semantics for them [21, 19, 25]. As pointed out by Abadi [2], 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 [22] 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. [21] 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 [19] 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 [73] 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 [94]. In [84, 15] checks are discussed informally and thus they do not automate this process. Besides, same as in [22] 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 [19] 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 [45, 84, 7, 50], 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 [21], 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, $\langle\{\{N_B\}_{K_{BS}}, x\}, [N_B \cdot K_{BS}/x]\rangle \triangleright N_B \cdot K_{BS}$. This is because $\mathtt{senc}(\mathtt{fst}(x), \mathtt{snd}(x))\sigma =_{E_{dy}} \{N_B\}_{K_{BS}}$ holds if and only if $x =_{E_{dy}} N_B \cdot K_{BS}$. This is usually referred as the "perfect encryption" assumption [6]. On the other hand, by the definition of $v_D(M)$ in [21], 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, which formalizes a principal's ability and 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 [4, 3] is similar in spirit to our operational equivalence (on knowledge states, Definition 3.5.1). 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 [25], 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 allows equality checks, it does not support implementation refinement, as we do here.

6.7 Conclusion and Future Work

In this work, 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 6.2 are not answered:

(i). Under what conditions does there exist a constraint base of a knowledge state?

(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.

CHAPTER 7: OFFLINE GUESSING ATTACKS

Although various past efforts have been made to characterize and detect guessing attacks, there is no consensus on the definition of guessing attacks. Such a lack of generic definition makes it extremely difficult to evaluate the resilience of security protocols to guessing attacks.

To overcome this hurdle, we seek a new definition in this thesis to fully characterize the attacker's guessing capabilities (i.e., guessability). This provides a general framework to reason about guessing attacks in a symbolic setting, independent of specific intruder models. We show how the framework can be used to analyze both passive and active guessing attacks.

Most of the results presented in this chapter are reported in our previous paper [74].

7.1 Introduction

Many security protocols are vulnerable to guessing attacks, which aim to obtain a poorly chosen password or data by trying every possible value for it. Let us reconsider the following simple one-way authentication protocol:

$$\text{Message 1.} \quad A \to B : \{N_A\}_{K_{AB}}$$

$$\text{Message 2.} \quad B \to A : \{\mathtt{f}(N_A)\}_{K_{AB}}$$

Here N_A is a fresh nonce generated by A and K_{AB} is the symmetric key shared between A and B, and $\mathtt{f}$ is a given function (e.g., $\mathtt{f}(N_A) = N_A + 1$). An attacker may obtain K_{AB} by trying to decrypt both messages with a guessed key k and then to compare the results, say r_1 and r_2: if r_2 equals $\mathtt{f}(r_1)$, then k is the correct guess. Such attacks become more feasible when one chooses a low entropy secret.

Starting from the early work of Gong et al. [56, 55], a lot of efforts have been made either to formulate guessing attacks or to detect them. Many approaches focus on heuristics to explore ways of validating a guess [35, 85, 59]. This is usually done by enumerating rules to determine whether a guess can be "verified", a term widely accepted to characterize a correct guess. These rules are used to derive an inference system modeling the guessing capabilities [44], by extending the standard Dolev-Yao model [48]. Realizing the "incompleteness" of such an inference system in a sense that it may fail to capture some guessing attacks, Drielsma et al. [49] develop a precise formalization of off-line guessing attacks, which is independent of any particular intruder model. However, no automatic procedure is given in [49] and, more importantly, it only allows guessing atomic values. In [35], Corin et al. first use static equivalence from the applied pi calculus [4] to characterize guessing attacks, which is then used to derive a procedure for detecting guessing attacks [10]. More recently, Blanchet and Abadi [14] refine the definition by imposing the observational equivalence condition.

Up to now, there is still no clear consensus regarding the general definition of guessing attacks, which explains why some protocol previously shown resistant to guessing attacks turns out to be vulnerable [76, 56]. There are two main reasons for this lack of generality.

First, the term “verifiable” is not fully understood or formalized, while being used implicitly as a synonym for “guessable” in all previous approaches. It is fair to mention here that several definitions regarding verifiability do exist, although none of them is general enough to be independent of protocol modeling and/or specific intruder models. For instance, Lowe [85] presents a group of rules to verify a guess. Indeed, these rules correctly identify verifiable guesses. It is unclear whether or not the rule set can completely cover all guesses that can actually be verified somehow, even under the Dolev-Yao intruder model. Similarly, Corin et al. [35] define a “verifiable” guess based on two conditions of a “verifier”. However, without any intuitive appeal, this definition can fail to capture some practically verifiable guess. Besides, the verifier itself can be very difficult to find. Corin et al. [33] then formulate a new definition of verifying a guess using static equivalence [4], which elegantly captures the essence of verifying a guess. Nonetheless, this definition may require the modeling of security protocols by the applied pi calculus. Moreover, it only considers guesses of atomic messages.

Second, guessing attacks have been studied from two different perspectives: (1) the process perspective [33, 10, 14], which relies on the modeling of security protocols; and (2) the attacker’s perspective [35, 44, 85], which emphasizes the guessing capabilities from a logical point of view. Neither provides a unified view towards guessing attacks.

This work is therefore geared towards a unified framework for the study of guessing attacks. The primary goal is to establish an intimate understanding of guessing, which is intuitive, yet provides a rigorous basis for guessing attacks. In other words, the new framework should be

- *faithful* (i.e., fits the common sense of guessing attacks),
- *expressive* (i.e., accounts for multiple guesses), and
- *complete* (i.e., captures all guessing attacks in a symbolic setting).

Unlike most previous work, we treat “guessing” and “attack” separately, because guessing relates closely to the attacker’s ability to reason about its knowledge, whereas attack further exploits the vulnerability of security protocols. It is worthwhile to reveal the dominant factor of a guessing attack — the attacker’s guessing capabilities or the interactions between entities.

7.1.1 Contributions

In this chapter, we propose a new definition to fully characterize the attacker’s guessing capabilities and then show how it relates to finding guessing attacks in security protocols. Specifically,

- To uncover relationship between “verifiable” and “guessable”, we formalize the idea of verifying a message in terms of recognizability [72] — the ability to distinguish a message from noise. To our best knowledge, this is the first definition of verifiability that is independent of security protocols and/or intruder models. We show, surprisingly, that a guessable message needs NOT to be verifiable. In other words, even though some message is not verifiable, it can still be guessed correctly by the attacker.
- We propose a weaker notion of verifiability to recover the intuitive understanding of guessing — a message can be guessed if and only if it is weakly verifiable.

This weaker notion thus provides a faithful, expressive, and complete framework for the study of guessing attacks.

- We introduce a novel way to evaluate the computational difficulty of guessing. While some guessing attack turns out to be (computationally) infeasible, the new metric provides an accurate way to discriminate between feasible and infeasible guessing attacks, reducing the gap between formal methods and real implementation. To our best knowledge, this is the first explicit measurement about guessing.
- As a case study, we apply our methodology to find passive guessing attacks under the standard Dolev-Yao intruder model and discuss how to extend this methodology to analyze active attacks.

7.1.2 Organization

In Section 7.2, we formalize the idea of verifying a guess and explain why (strong) verifiability is not a necessary condition for guessing. After presenting a new knowledge model that accounts for the attacker's guessing capabilities in Section 7.3, we introduce a weaker notion of verifiability that fully characterizes guessing capabilities in Section 7.4. In Section 7.5, we present our metric to gauge the computational difficulty of guessing. In Section 7.6, we move our attention to finding guessing attacks.

7.2 Formalizing the Idea of Verifying a Guess

As mentioned in the introduction, although the intuitive idea of verifying a guess has been extensively used to analyze guessing attacks in security protocols, it has not been adequately formalized. The purpose of this section is to formalize the meaning of “verifying a guess”.

It is crucial to note that verifiability requires one to distinguish useful information (a correct guess) from noise — an ability that is independent of security protocols. For instance, as seen in the example in the introduction, the attacker who knows $\{N_A\}_{K_{AB}}$ and $\{\mathtt{f}(N_A)\}_{K_{AB}}$ can easily test whether a message g is the correct guess of K_{AB}. And the test can be done off-line by checking

$$\mathtt{sdec}(\{\mathtt{f}(N_A)\}_{K_{AB}}, g) \stackrel{?}{=}_{E_{dy}} \mathtt{f}(\mathtt{sdec}(\{N_A\}_{K_{AB}}, g))$$

Some may argue, however, that for more complicated protocols (e.g., simplified LGSN protocol [47]) the attacker do need to communicate with other parties to verify a guess. We adopt a cognitive point of view here: verifying a guess is a process of using its knowledge, whereas communication is a way for protocol participants to exchange knowledge.

It is desirable to formalize verifiability independent of intruder models and security protocols. Although our concern appears to be different from previous chapter on detecting type-flaw attacks, the methodology is exactly the same: using one's knowledge to distinguish a message from another. We also build our work on the concept of recognizability.

Example 16. Consider again the one-way authentication protocol presented in the introduction. Assume a passive attacker can eavesdrop on communication links and save all the messages. Then, we can use $T_0 = \{\{N_A\}_{K_{AB}}, \{\mathtt{f}(N_A)\}_{K_{AB}}\}$ to represent the attacker's explicit knowledge. Here and hereafter, whenever needed, we implicitly add the public unary function symbol $\mathtt{f}$ into the term algebra presented in Figure 1.

Suppose that the attacker wants to guess the value of N_A and we use variable x to signify the guess. Let $T = T_0 \cup \{x\}$, $\sigma_1 = [N_A/x]$, and $\sigma_2 = [N_B/x]$. Clearly, $x\sigma_1$ is a correct guess, but $x\sigma_2$ is not. Then, it can be shown that $\sigma_1 \approx_{E_{dy},T} \sigma_2$. In other words, the attacker is unable to check whether a guess (of N_A) is correct or not.

We now suppose that the attacker wants to guess the value of K_{AB}. Again, we use x to signify the guess, and let $\sigma_3 = [K_{AB}/x]$ and $\sigma_4 = [N_B/x]$. We choose

$$u =_s \mathtt{sdec}(\{\mathtt{f}(N_A)\}_{K_{AB}}, x)$$

$$v =_s \mathtt{f}(\mathtt{sdec}(\{N_A\}_{K_{AB}}, x))$$

Then,

$$u\sigma_3 =_s \mathtt{sdec}(\mathtt{f}(\{N_A)\}_{K_{AB}}, K_{AB})$$

$$v\sigma_3 =_s \mathtt{f}(\mathtt{sdec}(\{N_A\}_{K_{AB}}, K_{AB}))$$

$$u\sigma_4 =_s \mathtt{sdec}(\{\mathtt{f}(N_A)\}_{K_{AB}}, N_B)$$

$$v\sigma_4 =_s \mathtt{f}(\mathtt{sdec}(\{N_A\}_{K_{AB}}, N_B))$$

Consider now, $T \vdash \{u, v\}$, $u\sigma_3 =_{E_{dy}} v\sigma_3 =_{E_{dy}} \mathtt{f}(N_A)$, and $u\sigma_4 \neq_{E_{dy}} v\sigma_4$. By the definition of operational equivalence, we have $\sigma_1 \not\approx_{E_{dy},T} \sigma_2$.

In the above example, we see that the attacker can discriminate a correct guess of K_{AB} from N_A by investigating the operational equivalence relation between two guesses (described by two substitutions): if the two different substitutions (resp. a correct and an incorrect guess) do not satisfy operational equivalence, then the guess can be verified; otherwise, the attacker cannot capture any nuance and the guess is not verifiable.

With this hindsight, we say a guess of t is *(strongly) verifiable* by T under equational theory E if $T \rhd t$ (i.e., t is recognizable by $\langle E, T, \epsilon \rangle$). This coincides with our intention of proposing the notion of recognizability. As in the previous example, we have $T \not\triangleright N_A$ and $T \triangleright K_{AB}$, which confirm that the protocol is vulnerable to off-line guessing attack.

Example 17. We extend the equational theory E_{dy} to model probabilistic encryption scheme by adding two public function symbols `renc` and `rdec`, and the following two equations:

$$\mathtt{rdec}(\mathtt{renc}(x, y, r), \mathtt{kp}(y)) = x$$

$$\mathtt{rdec}(\mathtt{renc}(x, \mathtt{kp}(y), r), y) = x$$

The new obtained equational theory E_{dyr} is as follows.

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

Figure 6: Equational Theory E_{dyr}.

Similar as $\{s\}_t$, we use $\{s\}_t^r$ to denote $\mathtt{renc}(s, t, r)$.

Let us consider the Encrypted Password Transmission (EPT) protocol [62]

$$\text{Message 1.} \quad S \to U : N_S \cdot K_S^+$$

$$\text{Message 2.} \quad U \to S : \{N_S \cdot P\}^r_{K_S^+}$$

Here, we use P to denote the secret password memorized by the user U and shared with the server S[4]. Now, suppose that a passive attacker explicitly knows N_S, K_S^+,

[4]In implementation, the secret password is either stored in plain text or hashed under some

and wants to guess P. Then, the attacker's knowledge state is $\langle E_{dry}, T, \epsilon\rangle$, where $T = \{N_S, K_S^+, \{N_S \cdot P\}^r_{K_S^+}\}$. Let $\sigma = [P/x]$ and $\sigma' = [P'/x]$, where $P \neq_{E_{dyr}} P'$. Here, we use σ and σ' to represent a correct and incorrect guesses of P, respectively.

Since the encryption scheme is randomized, the attacker does not know r and thus it is not able to compute $\{N_S \cdot P\}^r_{K_S^+}$ by the guess of P, say P'. It is not hard to see that for all u, v such that $T \cup \{x\} \vdash \{u, v\}$ we have $u\sigma_0 =_{E_{dyr}} v\sigma_0$ if and only if $u =_{E_{dyr}} v$. Similarly, for all u, v such that $T \cup \{x\} \vdash \{u, v\}$ we have $u[P'/x] =_{E_{dyr}} v[P'/x]$ if and only if $u =_{E_{dyr}} v$. Hence, $u\sigma_0 =_{E_{dyr}} v\sigma_0$ if and only if $u[P'/x] =_E v[P'/x]$. Because $\sigma_0 =_{E_{dyr}} [P'/x]$ needs not to be true, using the definition of recognizability we get $T \not\triangleright P$. This confirms the claim that this protocol is resistant to guessing attacks [62, 33].

However, if the protocol uses deterministic encryption, that is the second message is replaced by $\{N_S \cdot P\}_{K_S^+}$, then the value of P can actually be guessed. Let $T' = \{N_S, K_S^+, \{N_S \cdot P\}_{K_S^+}\}$. Towards a contradiction, suppose that $\sigma \approx_{E_{dy}, T\cup\{x\}} \sigma_0$ and $\sigma \neq_{E_{dy}} \sigma_0$.

Let $u =_s \{N_S \cdot x\}_{K_S^+}$ and $v = \{N_S \cdot P\}_{K_S^+}$. Clearly, $T \cup \{x\} \vdash \{u, v\}$ and $u\sigma_0 =_E v\sigma_0$. By the definition of operational equivalence, we get $u\sigma =_{E_{dy}} v\sigma$. That is, $\{N_S \cdot P'\}_{K_S^+} =_{E_{dy}} \{N_S \cdot P\}_{K_S^+}$. So, $P' =_{E_{dy}} P$ and thus $\sigma =_{E_{dy}} \sigma_0$, a contradiction. Therefore, $\sigma \approx_{E_{dy}, T\cup\{x\}} \sigma_0$ implies $\sigma =_{E_{dy}} \sigma_0$ and thus $T' \triangleright P$.

Indeed, (strong) verifiability implies the ability to guess. Nonetheless, we claim that this notion may fail to fully capture all possible guesses. Here's an example to show why.

---

one-way function.

Example 18. Let $T = \{N_A, \{N_A \cdot P\}_{K_B^+}\}$ denotes the attacker's explicit knowledge. Suppose that the attacker wants to guess the value of P, say P'. Note that the attacker does not know K_B^-. It is not hard to see that for all u, v such that $T \cup \{x\} \vdash \{u, v\}$ we have $u\sigma =_{E_{dy}} v\sigma$ if and only if $u =_{E_{dy}} v$. So, $u[P'/x] =_{E_{dy}} v[P'/x]$ if and only if $u[P/x] =_{E_{dy}} v[P/x]$. Since $P' =_{E_{dy}} P$ does not necessarily need to be true, using the definition of recognizability we know $T \not\triangleright P$. In other words, P is not strongly verifiable by T under E_{dy}.

Now, we suppose that the attacker first tries to guess K_B^-. Let $\sigma_0 = [K_B^-/x]$. Towards a contradiction, suppose that $\sigma \approx_{E_{dy}, T\cup\{x\}} \sigma_0$ and $\sigma \neq_{E_{dy}} \sigma_0$. Let $u =_s$ $\texttt{fst}(\texttt{sdec}(\{N_A \cdot P\}_{K_B^+}, x))$ and $v =_s N_A$. Clearly, $T \cup \{x\} \vdash \{u, v\}$ and $u\sigma_0 =_E v\sigma_0$. By the definition of operational equivalence, we get $u\sigma =_E v\sigma$. That is, $\texttt{fst}(\texttt{sdec}(\{N_A \cdot P\}_{K_B^+}, x))\sigma =_{E_{dy}} N_A$. So, $\sigma =_{E_{dy}} \sigma_0$, a contradiction. Therefore, $\sigma \approx_{E_{dy}, T\cup\{x\}} \sigma_0$ implies $\sigma =_{E_{dy}} \sigma_0$ and thus $T \triangleright E_{dy} K_B^-$. Then, with the correct guess of K_B^-, the attacker can easily get P.

We thus close this section by remarking that a complete characterization of guessing attacks requires a more general notion than strong verifiability.

7.3 Accounting for the Attacker's Guessing Capabilities

7.3.1 Explicit Guesses and Implicit Guesses

We have already seen in Example 18 that a guessable term is not necessarily a term that the attacker actually guesses. To avoid confusion, we use "explicit guess" to refer to the actual guess that the attacker makes; and "implicit guess" to refer to new terms deducible from the attacker's updated knowledge (i.e., knowledge plus explicit guess(es)). Besides, when we say a term is "guessable" or "can be guessed", we always refer to implicit guess. In this terminology, we say P is guessable by making explicit guess of K_B^- in Example 18. We tend to omit "implicit" or "explicit" when it is clear from the context.

As we will see, such a distinction between explicit and implicit guesses is important to understand the innate nature of guessing attacks. Let us consider some other examples that highlight this distinction.

Example 19. Let $T = \{N_A, K_B^+, \{N_A \cdot P\}_{K_B^+}\}$ denotes the attacker's explicit knowledge. Suppose that the attacker aims to obtain P. There are two possible ways: First, the attacker can explicitly guess P by using

$$\{N_A \cdot x\}_{K_B^+}\sigma =_{E_{dy}} \{N_A \cdot P\}_{K_B^+}$$

Second, it can explicitly guess K_B^- by using

$$\mathtt{fst}(\mathtt{pdec}(\{N_A \cdot P\}_{K_B^+}, y))\sigma =_{E_{dy}} N_A$$

These two methods differ in their explicit guesses. Clearly, the one with the shorter binary length is easier to be guessed.

The above example shows that to launch a guessing attack, there might be several ways for the attacker to make explicit guess. The following example illustrates the situation involves multiple explicit guesses.

Example 20. Let $T = \{N_A, K_B^+, \{N_A \cdot K_{AB}\}_{K_A^+}, \{N_A \cdot \{P\}_{K_{AB}}\}_{K_B^+}\}$ denotes the attacker's knowledge. Suppose that the attacker aims to obtain P (i.e., implicitly guess P). One straightforward way is by explicitly guessing K_A^- and P. Let x and y signify the two guesses, respectively. At first, the attacker can use

$$\mathtt{fst}(\mathtt{pdec}(\{N_A \cdot P\}_{K_B^+}, x))\sigma =_{E_{dy}} N_A$$

to obtain the correct guess of K_A^-. Then, it gets K_{AB} by decrypting $\{N_A \cdot K_{AB}\}_{K_A^+}$. Finally, it can use

$$\{N_A \cdot \{y\}_{K_{AB}}\}_{K_B^+}\sigma =_{E_{dy}} \{N_A \cdot \{P\}_{K_{AB}}\}_{K_B^+}$$

to obtain the correct guess of P.

We close this subsection by remarking that an explicit guess might turn out to be an implicit one, due to the redundancy in explicit guesses. For example, suppose the attacker knows $\{N_A, \{N_A \cdot P\}_{K_{AS}}\}$ and it makes explicit guesses of K_{AS} and P. Note that

$$\texttt{snd}(\texttt{sdec}(\{N_A \cdot P\}_{K_{AS}}, K_{AS})) =_{E_{dy}} P$$

It is not hard to see that P can be derived from the explicit guess of K_{AS}. So, there is no need to make explicit guess of P. We postpone to Section 7.5 some further discussion of the redundancy in explicit guesses.

7.4 A Complete Characterization of Guessing

In this section, we introduce a weaker notion of verifiability to fully characterize the intuitive understanding of guessing.

The possible-worlds semantics lends more sense to recognizability: a term t (indicated by x) is recognizable if and only if x indicates t (i.e., $x\sigma =_E t$) in all possible states. This suggests that $T \rhd t$ is insufficient for the case of multiple free variables (indicating potentially ambiguous messages or unchecked guesses).

However, a closer look at the original definition of recognizability (Definition 3.5.1) shows that there are two types of free variables. For convenience, we repeat the definition here.

Definition 3.5.1 (Recognizability). *Let* $\vec{T} = \langle E, T, \sigma\rangle$ *be one's knowledge state and* t *be a potentially ambiguous message (denoted by* z*). Then, we say that* t *is* recognizable *by* $\vec{T}$ *and write* $\vec{T} \rhd t$*, if and only if* $\langle E, T \cup \{z\}, \sigma[t/z]\rangle \models \mathbf{K}\boldsymbol{dicto}(z)$*.*

Clearly, the first variable type contains only the variable z and yet the second type of variables are those occurs in T (i.e., $fv(T)$).

Example 21. We continue with Example 20. The attacker's knowledge state is represented by

$$\vec{T} = \langle\{N_A, \{N_A \cdot K_{AB}\}_{K_A^+}, \{N_A \cdot \{P\}_{K_{AB}}\}_{K_B^+}, x, y\}, [K_B^-/x, P/y]\rangle$$

in which x and y correspond to two distinct explicit guesses made by the attacker. Then, $\vec{T} \rhd P$. However, if the attack only makes a single guess, either K_B^- or P, then $\vec{T}' \not\rhd P$, where $\vec{T}'$ is either

$$\langle\{N_A, \{N_A \cdot K_{AB}\}_{K_A^+}, \{N_A \cdot \{P\}_{K_{AB}}\}_{K_B^+}, x\}, [K_B^-/x]\rangle$$

or

$$\langle\{N_A, \{N_A \cdot K_{AB}\}_{K_A^+}, \{N_A \cdot \{P\}_{K_{AB}}\}_{K_B^+}, y\}, [P/y]\rangle$$

At this point, one may be tempted to conjecture that this more general notion of recognizability suffices to describe the desired new notion of verifiability. Unfortunately, this is not the case, because in Definition 3.5.1 $[t/z]$ is composed with σ, introducing a new explicit guess of t, as shown by the following example.

Example 22. Let $\vec{T} = \langle\{N_A, \{(N_A \cdot N_B) \cdot \{N_A\}_{K_B^+}\}_{K_{AS}}, x\}, [K_{AS}/x]\rangle$ denotes the attacker's knowledge. Suppose that the attacker wants to obtain K_B^+. Note that the attacker only makes one explicit guess of K_{AS}. It is not hard to see that the attacker indeed can correctly guess K_{AS}. Then, the attacker's knowledge becomes $\vec{T}' = \langle\{N_A, N_B, K_{AS}, \{N_A\}_{K_B^+}\}, \epsilon\rangle$. Now, it is not hard to see that, without any further guess(es), the attacker is still not able to obtain K_B^+. On the other hand, however, it can be shown that $\vec{T} \rhd K_B^+$.

There is one simple fix to avoid adding the new explicit guess. As explained earlier, an explicit guess may turn out to be an implicit one by exploiting the redundancy in explicit guesses. The trick is that we impose condition(s) to ensure that the newly added explicit guess becomes an implicit one.

Definition 7.4.1 (Weak Verifiability). Let $\vec{T} = \langle E, T, \sigma_0 \rangle$ be a knowledge state and t be a ground term. We say that t is *weakly verifiable* by $\vec{T}$ and write $\vec{T} \blacktriangleright t$ if $\vec{T} \rhd t$ and $T\sigma_0 \vdash_E t$.

The condition $T\sigma_0 \vdash_E t$ implies that $T \vdash s$ and $s\sigma_0 =_E t$ for some s. In other words, the explicit guess can be exactly described by using T, obviating the need to explicitly guess t. The following lemma states this formally.

Lemma 7.4.2. Let $\vec{T} = \langle E, T, \sigma_0 \rangle$ be a knowledge state and t be a ground term. If $\vec{T} \blacktriangleright t$, then there exists a term s such that $T \vdash s$ and $s\sigma_0 =_E s\sigma =_E t$ for all $\sigma \approx_{E,T} \sigma_0$.

Proof. By Definition 7.4.1, we have $\vec{T} \rhd t$ and $T\sigma_0 \rhd t$. Then, it follows from Lemma 2.1.6 that there exists a term s such that $T \vdash s$ and $s\sigma_0 =_E t$. It remains to show that $s\sigma =_E t$ for all $\sigma \approx_{E,T} \sigma_0$.

Let $s\sigma =_E t'$ and x be a fresh variable. Since $\sigma \approx_{E,T} \sigma_0$, we get $\sigma \circ [t'/x] \approx_{E,T\cup\{x\}} \sigma_0 \circ [t/x]$. Moreover, since $\vec{T} \rhd t$, we thus have $x\sigma \circ [t'/x] =_E t$ by Definition 3.3.1. Hence, $t' =_E t$. This completes the proof. $\square$

Recall the example given at the end of Section 7.3.1, where the attacker knows N_A and $\{N_A \cdot P\}_{K_{AS}}$. Suppose that it only makes one explicit guess of K_{AS} and aims to obtain P. Then, his knowledge is represented by

$$\vec{T} = \langle \{N_A, \{N_A \cdot P\}_{K_{AS}}, x\}, [K_{AS}/x] \rangle$$

Moreover, it can be shown that $\vec{T} \rhd E_{dy} P$ and $T[K_{AS}/x] \vdash_{E_{dy}} P$. That is, P is weakly verifiable by $\vec{T}$. Here, the attacker needs not to explicitly guess P.

On the contrary, in Example 22, we notice that

$$\{N_A, \{(N_A \cdot N_B) \cdot \{N_A\}_{K_B^+}\}_{K_{AS}}, x\}[K_{AS}/x] \nvdash_{E_{dy}} K_B^+$$

Thus, as noted before, the attacker has to make other explicit guess(es) (e.g., a guess of K_B^+) to obtain K_B^+.

7.4.1 Guessability

Finally, we coin the term guessability (i.e., the attacker's ability to guess) in terms of weak verifiability.

Definition 7.4.3 (Guessability). Let $\vec{T}$ be one's knowledge state. Then, a ground term t is *guessable* if and only if $\vec{T} \blacktriangleright t$.

This provides the last step to formalize and justify the long held intuition between "guess" and "verify".

Noticing that the attacker's knowledge should be updated to $\langle T \cup \{t\}, \sigma \rangle$ if $\langle T, \sigma \rangle \blacktriangleright_E t$, one may reasonably think that we need to recursively add new guessable terms into the attacker's knowledge until no new guessable term can be found. It seems probable that Definition 7.4.3 fails to account for this dynamics.

Somewhat surprisingly, we find that adding t into the attacker's knowledge makes no difference in terms of guessability. The following theorem states this formally and justifies the Definition 7.4.3.

Theorem 7.4.4. Suppose that $\langle E, T, \sigma_0 \rangle \blacktriangleright s$. Then, $\langle E, T, \sigma_0 \rangle \blacktriangleright t$ if and only if $\langle E, T \cup \{s\}, \sigma_0 \rangle \blacktriangleright t$.

7.5 The Difficulty of Guessing

Until now we have mainly focused on the possibility of guessing. In this section, we concern ourselves with the difficulty of guessing, that is, how much computational efforts are required to obtain a guessable term t, provided $\vec{T} \blacktriangleright t$.

It should be noted that different guessing problems incur different computational cost. For example, (explicitly) guessing a 128-bit symmetric key is significantly harder than guessing a poorly chosen password. In fact, there is a physical argument [71] that implies that guessing a 128-bit symmetric key is “practically infeasible”. Moreover, even for the same guessing problem, the efforts can vary considerably in different ways of (explicit) guessing. For instance, in Example 19, the attacker can either explicitly guess P or explicitly guess K_B^- to obtain P. Let us assume K_B^- is a 1024-bit private key and P is a poorly chosen password. Then, guessing P could be much easier than guessing K_B^-.

Thus, despite the guessability results, we also need a new notion to characterize the difficulty of guessing. One may think of using the binary length of all the explicit guesses. Unfortunately, this simple way may fail to faithfully characterize the difficulty, as the following examples show.

Example 23. Let us consider two scenarios, in which the attacker's knowledge state is, respectively, represented by

$$\vec{T_1} = \langle \{N_A, \{N_A \cdot P\}_{K_{AB}}, \{N_A \cdot K_A^+\}_{K_{AS}}\}, x, y\},$$
$$[K_{AB}/x, K_{AS}/y]\rangle$$

and

$$\vec{T_2} = \langle \{N_A, \{\{N_A \cdot P\}_{K_B^+}\}_{K_{AB}}, \{K_B^-\}_{K_{AS}}, x, y\},$$
$$[K_{AB}/x, K_{AS}/y]\rangle$$

Suppose that the attacker wants to obtain $\{P\}_{K_A^+}$ in the first scenario and P in the second. In both cases, these can be done by explicitly guessing K_{AB} and K_{AS}. It is tempting to conclude that guessing $\{P\}_{K_A^+}$ and P is equally difficult.

However, a closer examination reveals the difference.

In the first scenario, the attacker can use

$$\texttt{fst}(\texttt{sdec}(\{N_A \cdot P\}_{K_{AB}}, x))\sigma =_{E_{dy}} N_A \tag{16}$$

to obtain the correct guess of K_{AB}. Note that Equation (16) does not involve the guess of K_{AS}. So, the attacker can correctly guess K_{AB} without guessing K_{AS}. Similarly, we see that the attacker can also correctly guess K_{AS} without guessing K_{AB}. After correctly guessing K_{AB} and K_{AS}, the attacker can easily get P and K_A^+, and thus derive $\{P\}_{K_A^+}$. To sum up, the maximum number of times the attacker has attempted to obtain $\{P\}_{K_A^+}$ is $2^{|K_{AB}|} + 2^{|K_{AS}|}$.

On the contrary, in the second scenario, the attacker can only use

$$\texttt{fst}(\texttt{sdec}(\texttt{sdec}(\{\{N_A \cdot P\}_{K_B^+}\}_{K_{AB}}, x), \texttt{pdec}(\{K_B^-\}_{K_{AS}}, y)))\sigma =_{E_{dy}} N_A \tag{17}$$

to obtain the correct guesses of K_{AB} and K_{AS}, and thus derive P. This means the attacker has to guess K_{AB} and K_{AS} simultaneously. Hence, the maximum number of times it has attempted to obtain P is $2^{|K_{AB}|+|K_{AS}|}$.

Therefore, guessing in the second scenario is considerably harder than in the first scenario.

Example 24. Let

$$\vec{T} = \langle \{N_A, \{N_A \cdot P\}_{K_{AB}}, \{K_{AS}\}_P, \{N_A \cdot K_B^+\}_{K_{AS}}, x, y\}, \\ [K_{AB}/x, K_{AS}/y] \rangle$$

denotes the attacker's knowledge state. Suppose that the attacker wants to obtain $\{P\}_{K_B^+}$. Similar to the first scenario in the previous example both explicit guesses (of K_{AB} and K_{AS}) can be made independently. But we have to be careful not to conclude that the maximum number of times the attacker has attempted to obtain $\{P\}_{K_B^+}$ is also $2^{|K_{AB}|+|K_{AS}|}$.

Let us take a closer look at $\vec{T}$. We notice that after obtaining the correct guess of K_{AB} the attacker can use $\texttt{snd}(\texttt{sdec}(\{N_A \cdot P\}_{K_{AB}}, K_{AB})) =_{E_{dy}} P$ to derive P, which can be further used to derive K_{AS} as $\texttt{sdec}(\{K_{AS}\}_P, P) =_{E_{dy}} K_{AS}$. So, the attacker can derive K_{AS} only by a single explicit guess of K_{AB}. In other words, the maximum number of times the attacker has attempted is just $2^{|K_{AB}|}$.

As noted in the above examples, the number of bits that the attacker has to guess might be less than that of all explicit guesses. There are two main reasons for this: (i) some explicit guess(es) can be readily made without dealing with other guesses, dividing an overall hard guess problem into several easier ones; and (ii) the redundancy inherent in all the explicit guesses makes it possible to derive useful information between them.

We thus propose to use the search space, rather than the number of bits of the explicit guesses, to characterize the difficulty of guess.

Definition 7.5.1 (Computational Difficulty). We define $minmax(\vec{T} \blacktriangleright t)$ as the minimum maximum number of times one might attempt to obtain t. Moreover, we say that the *computational difficulty* of $\vec{T} \blacktriangleright t$ is *in order of* n (or n*-bit hard*) if $n = \lceil \log_2 minmax(\vec{T} \blacktriangleright t) \rceil$.

Now, it is not hard to see that $\vec{T_1} \blacktriangleright \{P\}_{K_A^+}$ and $\vec{T_2} \blacktriangleright P$ in Example 23 are in order of $\log_2 (2^{|K_{AB}|} + 2^{|K_{AS}|})$ and $|K_{AB}| + |K_{AS}|$, respectively; $\vec{T} \blacktriangleright \{P\}_{K_B^+}$ in Example 24 is in order of $|K_{AB}|$.

Although Definition 7.5.1 allows us to evaluate the difficulty of guess accurately, it does not provide much insight into how to determine $minmax(\vec{T} \blacktriangleright t)$ and thus the difficulty of $\vec{T} \blacktriangleright t$. Obviously, much future work remains to be done for solving $minmax(\vec{T} \blacktriangleright t)$. There are two issues to be considered in addressing this problem: first, to explore the redundancy in those explicit guesses, and second, to partition the explicit guesses into groups that can be done without involving others. We do not explore these issues further here.

7.6 Detecting Guessing Attacks

In this section, we briefly discuss how the proposed framework can be used effectively in detecting guessing attacks.

7.6.1 A Cognitive Perspective

Before diving into the technical discussion, it helps to have a clear distinction between passive and active attacks (not just guessing attacks).

7.6.2 Passive Attacks

The passive attacker does not interact with protocol participants; whether or not it can launch an attack solely based upon the eavesdropped data. We thus informally view the passive attack as a *computing* problem: given a set of observed messages, whether it is possible to "compute" confidential data.

In the literature, intruder deduction [32, 3, 42, 36] and static equivalence [4, 3, 14, 27] correspond to this computational view, where computing is regarded as a knowledge reasoning process.

7.6.3 Active Attacks

Besides its ability to reason about knowledge as the passive attacker, the active attacker can also communicate with legitimate participants. Benefit from a cognitive perspective, this can be understood in two complementary ways:

1 (Communication view) we can think of communication with external entities as a way of gaining new information that cannot be deduced from its current knowledge.

2 (Computational view) we can regards the external entities as as an internal oracle that computes new information from its current knowledge.

Example 25. Let us consider again the protocol presented in the introduction:

$$\text{Message 1.} \quad A \to B : \{N_A\}_{K_{AB}}$$

$$\text{Message 2.} \quad B \to A : \{\mathtt{f}(N_A)\}_{K_{AB}}$$

An active attacker can act in the role of A initiate communication with B. Assume that the attacker's explicit knowledge is represented by term set $T_I = \{I, A, B, \{N_A\}_{K_{AB}}\}$.

From a communication point of view, the attacker does not know $\{\mathtt{f}(N_A)\}_{K_{AB}}$ (i.e., $T_I \nvdash_{E_{dy}} \{\mathtt{f}(N_A)\}_{K_{AB}}$) at first. Only after exchanging messages with B, it obtain message $\{\mathtt{f}(N_A)\}_{K_{AB}}$ and thus its explicit knowledge becomes

$$T'_I = \{I, A, B, \{N_A\}_{K_{AB}}, \{\mathtt{f}(N_A)\}_{K_{AB}}\}$$

Clearly,

$$T_I \not\equiv_{E_{dy}} T'_I \tag{18}$$

From a computational point of view, the attacker is endowed with an oracle that takes t as input and outputs

$$\mathtt{g}(t) = \mathtt{senc}(\mathtt{f}(\mathtt{sdec}(t, K_{AB})), K_{AB}) \tag{19}$$

where $\mathtt{g}$ is a public function symbol that never occurs in the original term algebra $\mathcal{T}$. As the oracle is internal, we thus incorporate the above equation to equation theory E_{dy} and get E'_{dy}. Therefore,

$$T_I \equiv_{E'_{dy}} T'_I \tag{20}$$

In this light, we can categorize the security protocol models into two groups: one is based on communication view, such as Strand Space Model [52], CSP [101], and applied pi-calculus [4]; the other is based on computational view, such as multiset rewriting [24], constraint solving[92], Prolog rules [12], and Horn clauses [13].

We remark that a clear distinction between passive and active attack enables us to determine whether the attack is primarily due to the attacker's knowledge or its interaction with legitimate participants. Moreover, a thorough understanding of passive attacks will shed important light on the study of active attacks and security protocol design as well.

7.6.4 Passive Guessing Attacks

In terms of passive guessing attack, the knowledge reasoning problem is that, given a set of observed messages, whether it is at all possible to correctly guess any confidential data.

Our framework formulates the above knowledge reasoning problem accurately. We use term set T to describe the set of observed messages, term t to represent some confidential data, variables set X to correspond to all the guess made by the attacker, and substitution σ with $Dom(\sigma) = X$ to indicate the correct guesses. Because passive eavesdropping is performed over legitimate protocol sessions, observed messages must comply with the protocol specification and thus we can assume T to be a ground term set. Likewise, t is also ground. Then, $\langle E, T \cup X, \sigma\rangle$ models the passive attacker's knowledge state. Finally, the problem of detecting passive guessing attacks is reduced to deciding $\langle E, T \cup X, \sigma\rangle \blacktriangleright t$.

At this point, detection of passive guessing attacks boils down to deciding guessability. The last missing step is to give a decision procedure for $\langle E, T \cup X, \sigma\rangle \blacktriangleright t$. Unfortunately, in general, this may be undecidable [3].

7.6.5 Deciding Guessability under standard Dolev-Yao intruder model

In part II, we propose a terminating procedure to determine recognizability under standard Dolev-Yao intruder model [48]. Here, we adopt this procedure to decide guessability under Dolev-Yao model.

Although the original procedure (i.e., algorithm *solve*) is intended for deciding recognizability, it can be easily extended to decide guessability.

Theorem 7.6.1. Let $\langle T, \sigma \rangle$ be a knowledge state, t be a ground term, and x be a fresh variable. Suppose that $T\sigma \cup \{t\}$ does not contain function symbol `fst`, `snd`, `pdec`, or `sdec`. If $T\sigma \vdash_{E_{dy}} t$, $solve(\langle T \cup \{x\}, \epsilon, \sigma \circ [t/x] \rangle)$ returns $\langle T', \eta', \sigma' \rangle$, and $x\eta' =_s t$, then $\vec{T} \blacktriangleright t$.

Please refer to Chapter 5 for more details on the algorithm.

7.6.6 Extension to Active Guessing Attacks

To handle an active attacker, it is important to model security protocols. As mentioned in Section 7.6.1, existing formal methods for protocol modeling fall into two groups: communication based and computation based.

For simplicity, we adopt a computational view here: we regard the active attacker as a special passive attacker with an oracle. More specifically, we can add equations describing the oracle to the original equational theory. For instance in Example 25, we just add Equation 19 to equation theory E_{dy} (and obtain equational theory E'_{dy}). This method is similar to that of [10], which uses a set of second-order variables to keep track of the computations. In general, a symbolic trace [53, 18, 30] that describes the sequences of actions (receive or send) of a given protocol role brings about n distinct equations, where n is the number of messages sent by the role.

By extending the original equational theory, we get a new equational theory, say E', to model the active attacker's capabilities[5]. Therefore, the problem of detecting active guessing attack boils down to deciding guessability under the new equational theory E'.

[5]In fact, the original term algebra $\mathcal{T}$ is also extended to $\mathcal{T}'$, which includes several new public function symbols modeling the oracle computation.

It should be noted that deciding $\blacktriangleright$ under the new equational theory E' may be undecidable. After all, the our approach considers an unbounded number of sessions of the protocol [103, 26], for which protocol insecurity is undecidable [51]. Approximation techniques [40, 13] are usually employed to handle unbounded verification. Due to space limit, we do not pursue these further here.

7.6.7 Active guessing attack is passive guessing attack?

Thanks to the clear distinction between passive and active attack, we find surprisingly that in many cases the enhanced capabilities of active attacker does not impact guessability at all; that is to say, active attacker is no more powerful than passive attacker in term of guessability.

For example, in the protocol given at the beginning of the introduction, if an attacker knows $\{\{N_A\}_{K_{AB}}, \{\mathtt{f}(N_A)\}_{K_{AB}}\}$ and makes explicit guess of K_{AB}, then all actively guessable terms are actually passively guessable, as the following proposition shows.

Proposition 7.6.2. Let $\vec{T}$ and $\vec{T}'$ be two knowledge states and t be a ground term. Suppose that

$$\vec{T} = \langle E_{dy}, T, [K_{AB}/x] \rangle$$

$$\vec{T}' = \langle E'_{dy}, T, [K_{AB}/x] \rangle$$

$$T = \{\{N_A\}_{K_{AB}}, \{\mathtt{f}(N_A)\}_{K_{AB}}, x\}$$

and t does not contain function symbol $\mathtt{g}$, $\mathtt{sdec}$, $\mathtt{fst}$, or $\mathtt{snd}$. Then, $\vec{T} \blacktriangleright t$ if and only if $\vec{T}' \blacktriangleright t$.

To prove Proposition 7.6.2, we need the following lemma.

Lemma 7.6.3. Let $S = \{N_A, K_{AB}\}$. Suppose that $l \rightarrow r \in R_{E'_{dy}}$. If $S \vdash C[l\theta]$, then $S \vdash C[r\theta]$.

Proof. We make induction on $\|C\|$. For the base case, $\|C\| = 1$, a case by case analysis shows that $S \vdash r\theta$ if $l \rightarrow r \in R_{E_{dy}}$. Now, we consider the case when $l =_s \mathtt{g}(x)$ and $r =_s \mathtt{senc}(\mathtt{f}(\mathtt{sdec}(x, K_{AB})), K_{AB})$. Without loss of generality, let $\theta = [t/x]$. Then, $S \vdash \mathtt{g}(t)$ and thus $S \vdash t$. Since $C[r\theta] =_s r\theta =_s \mathtt{senc}(\mathtt{f}(\mathtt{sdec}(t, K_{AB})), K_{AB})$ and $S \vdash \{t, K_{AB}\}$, we have $S \vdash C[r\theta]$. Now, we suppose the claim holds for all $\|C\| \leq k$.

For $\|C\| = k + 1$, let $C =_s f(t_1, t_2, \cdots, t_n)$ where $t_j =_s C'[l\theta]$ for some context C'. Clearly, $S \vdash t_i$ for $1 \leq i \leq n$. By induction, we get $S \vdash C'[r\theta]$ and thus

$$S \vdash f(t_1, t_2, \cdots, C'[r\theta], \cdots, t_n) =_s C[r\theta]$$

This completes the proof. $\square$

Proof of Proposition 7.6.2. (Sketch) For simplicity, we let $T = \{\{N_A\}_{K_{AB}}, \{\mathtt{f}(N_A)\}_{K_{AB}}, x\}$, $\sigma_0 = [K_{AB}/x]$, $S = \{N_A, K_{AB}\}$, and $\eta = \sigma_0 \circ [t/y]$. Clearly, $T\sigma_0 \equiv_{E_{dy}(E'_{dy})} S$.

("If" part) Using the definition of guessability, we have $\vec{T} \rhd t$ and $T\sigma_0 \vdash_{E_{dy}} t$. Note that $E_{dy} \subset {E_{dy}}'$. So, $T\sigma_0 \vdash_{E'_{dy}} t$. Thus, to prove $\vec{T} \blacktriangleright_{E'_{dy}} t$ it remains to show that $\vec{T'} \rhd t$, that is, $y\sigma =_{E'_{dy}} t$ for all σ satisfying $\sigma \approx_{E'_{dy}, T \cup \{y\}} \eta$.

Let σ be an $R_{E'_{dy}}$-normal substitution satisfying $\sigma \approx_{E'_{dy}, T \cup \{y\}} \eta$. Then, by Definition 3.3.1, for all terms u and v such that $T \cup \{y\} \vdash \{u, v\}$ we have $u\sigma =_{E'_{dy}} v\sigma$ if and only if $u\eta =_{E'_{dy}} v\eta$. We further assume that neither u or v contains function symbol $\mathtt{g}$. Note that $Ran(\sigma)$ does not contain function symbol $\mathtt{g}$ either, because otherwise σ is not $R_{E'_{dy}}$-normal. It is not hard to see that $u\sigma =_{E'_{dy}} v\sigma$ if and only if $u\sigma =_{E_{dy}} v\sigma$; likewise, $u\eta =_{E'_{dy}} v\eta$ if and only if $u\eta =_{E_{dy}} v\eta$. Thus, for all terms terms u and v such that

- $T \cup \{y\} \vdash \{u, v\}$, and
- neither u or v contains function symbol g,

we have $u\sigma =_{E'_{dy}} v\sigma$ if and only if $u\eta =_{E'_{dy}} v\eta$. That is, $\sigma \approx_{E_{dy},T\cup\{y\}} \eta$. By assumption, $\vec{T} \rhd t$ and thus $y\sigma =_{E_{dy}} t$. Clearly, $y\sigma =_{E'_{dy}} t$, as required.

("Only if" part) Using the definition of guessability, we have $\vec{T'} \rhd t$ and $T\sigma_0 \vdash_{E'_{dy}} t$. To prove $\vec{T} \blacktriangleright t$, we need to show $\vec{T} \rhd t$ and $T\sigma_0 \vdash_{E_{dy}} t$.

(i). We show $T\sigma_0 \vdash_{E_{dy}} t$. By assumption, $T\sigma_0 \vdash_{E'_{dy}} t$ and thus $S \vdash_{E'_{dy}} t$. Using the definition of $\vdash_{E'_{dy}}$, we have $S \vdash s$ and $s \to^{!}_{R_{E'_{dy}}} t$ for some s. It follows from Lemma 7.6.3 that $S \vdash t$. Note that $S \equiv_{E_{dy}} T\sigma_0$ and $S \vdash_{E_{dy}} t$. We know that $T\sigma_0 \vdash_{E_{dy}} t$.

(ii). We show $\vec{T} \rhd t$, that is, $y\sigma =_{E_{dy}} t$ for all σ satisfying $\sigma \approx_{E_{dy},T\cup\{y\}} \eta$. Let σ be an arbitrary substitution satisfying $\sigma \approx_{E_{dy},T\cup\{y\}} \eta$. Then, by Definition 3.3.1 we know that $Dom(\sigma) = \{x, y\}$ and $u\sigma =_{E_{dy}} v\sigma$ if and only if $u\eta =_{E_{dy}} v\eta$ for all u, v such that $T \cup \{y\} \vdash \{u, v\}$. Note that $Ran(\sigma)$ does not contain function symbol g.

By $\vec{T} \rhd K_{AB}$, it can be shown that $x\sigma =_s K_{AB}$. Then, without loss of generality, we let u' and v' be two terms such that $T \cup \{y\} \vdash \{u', v'\}$ and both may contain function symbol g. We need to prove $u'\sigma =_{E'_{dy}} v'\sigma$ if and only if $u'\eta =_{E'_{dy}} v'\eta$. Let n be the number of times the function symbol g occurs in u' and v'. We proceed by induction on n.

For the base case, $n = 0$, by assumption, we know that $u'\sigma =_{E_{dy}} v'\sigma$ if and only if $u'\eta =_{E_{dy}} v'\eta$. Now, we suppose that $u'\sigma =_{E'_{dy}} v'\sigma$ if and only if $u'\eta =_{E'_{dy}} v'\eta$ for all $n \leq k$.

For $n = k+1$, without loss of generality, we can let $u' =_s C[\mathsf{g}(w)]$ for some context C and term w. Since $T \cup \{y\} \vdash u'$ and $\mathsf{g}(w) \subseteq u'$, it can be shown that $T \cup \{y\} \vdash \mathsf{g}(w)$.

Note that $\mathtt{g}$ does not occur in $T \cup \{y\}$. So, $T \cup \{y\} \vdash w$.

Since $u' \to_{R_{E'_{dy}}} C[\mathtt{senc}(\mathtt{f}(\mathtt{sdec}(w, K_{AB})), K_{AB})]$, we have

$$\begin{aligned} u'\sigma &\to_{R_{E'_{dy}}} C[\mathtt{senc}(\mathtt{f}(\mathtt{sdec}(w, K_{AB})), K_{AB})]\sigma \\ &=_s C[\mathtt{senc}(\mathtt{f}(\mathtt{sdec}(w, x)), x)]\sigma \end{aligned} \tag{21}$$

Note that $T \cup \{y\} \vdash \{x, w\}$. It is clear that $T \cup \{y\} \vdash \mathtt{senc}(\mathtt{f}(\mathtt{sdec}(w, x)), x)$.

We also notice that $T \cup \{y\} \vdash C[\mathtt{g}(w)]$, $\mathtt{g}(w)$ does not occur in $T \cup \{y\}$, and $T \cup \{y\} \vdash \mathtt{senc}(\mathtt{f}(\mathtt{sdec}(w, x)), x)$. Thus, we obtain $T \cup \{y\} \vdash C[\mathtt{senc}(\mathtt{f}(\mathtt{sdec}(w, x)), x)]$. Let $u'' =_s C[\mathtt{senc}(\mathtt{f}(\mathtt{sdec}(w, x)), x)]$.

Consider now, $T \cup \{y\} \vdash \{u'', v'\}$ and the number of times $\mathtt{g}$ occurs in u'' and v' is k. By induction hypothesis, we have $u''\sigma =_{E'_{dy}} v'\sigma$ if and only if $u''\eta =_{E'_{dy}} v'\eta$. Moreover,

$$\begin{aligned} u'\eta =_s C[\mathtt{g}(w)]\eta &\to_{R_{E'_{dy}}} C[\mathtt{senc}(\mathtt{f}(\mathtt{sdec}(w, K_{AB})), K_{AB})]\eta \\ &=_s C[\mathtt{senc}(\mathtt{f}(\mathtt{sdec}(w, x)), x)]\eta =_s u''\eta \end{aligned} \tag{22}$$

Then, we know from (21) and (22)that $u'\sigma =_{E'_{dy}} u''\sigma$ and $u'\eta =_{E'_{dy}} u''\eta$. Thus, $u'\sigma =_{E'_{dy}} v'\sigma$ if and only if $u'\eta =_{E'_{dy}} v'\eta$.

By Definition 3.3.1, we get $\sigma \approx_{E'_{dy}, T \cup \{y\}} \eta$. By assumption, $\vec{T}' \rhd t$, we know from Definition 3.3.1 that $y\sigma =_{E'_{dy}} t$. Note that $Ran(\sigma)$ does not contain $\mathtt{g}$. Consequently, $y\sigma =_{E_{dy}} t$. Therefore, $\vec{T} \rhd t$, as required.

This completes the proof. $\square$

7.7 Conclusion

In this chapter, we present a general framework of guessing, which clarifies and formalizes the intuitive understanding of "verifying a guess". Thanks to its following innovative features

- independence of any specific adversary model,
- support of multiple (explicit) guesses, and
- definition to measure the computational difficulty of guessing

this framework enables us to detect passive and active guessing attacks, both of which rely critically on the decision problem $\rhd$.

Apart from the technical contributions of this chapter, other messages we want to convey are that passive attacks are as important as active attacks, especially in the study of guessing attacks; and that both communication and computational views of active attacks may offer new insight in security protocol analysis.

There are two major limitations of this study. First, the standard Dolev-Yao model considered in Section 7.6.4 assumes "perfect encryption", that is, $\{m\}_k =_{E_{dy}} \{m'\}_{k'}$ if and only if $m =_{E_{dy}} m'$ and $k =_{E_{dy}} k'$. Such an assumption is unrealistic for cryptographic primitives with visible algebraic properties such as exclusive or and homomorphic operator, see [38] for a survey. Second, our definition of computational difficulty is too general to be practically useful and it is non-trivial to determine $minmax(\vec{T} \blacktriangleright t)$. Moreover, our analysis in Example 23 and 24 assumes a uniform distribution of the guessing value and thus there is no better way than brute force guessing. However, in reality, weak secret (say, n bits) usually has low entropy, making it easier to guess ($< n$-bit hard).

Our future work will be aimed at addressing these limitations. In particular, we plan to investigate the problem of detecting guessability under more general equational theories and develop automatic tools to detect guessing attacks.

CHAPTER 8: CONCLUSIONS

In this thesis, we provide a satisfying answer to the question "What is meant by saying that a message can be verified" by proposing a third knowledge notion — recognizability — in security protocols. The notion of recognizability lies somewhere between deduction and static equivalence, which are two traditional knowledge notions in security protocols. A decision procedure is given to decide recognizability under standard Dolev-Yao intruder model. More importantly, the notion of recognizability is also applied in various security protocol analysis tasks.

The notion of recognizability is extended to elicit semantics of protocol narrations and thus a protocol compilation procedure. The new protocol compilation process achieves a consensus view of security protocols for protocol designer, protocol implementer, and even the attacker. Such a view is important in a sense that (i) it enables the protocol designer to realistically consider other possible protocol executions rather than the expected one, (ii) it ensures that protocol implementer to conduct all necessary internal checks in protocol implementations, and (iii) it provides a path to more secure protocol designs and implementations. Two types of attacks are identified as those can be thawed through adjusting the protocol implementation. In particular, type-flaw attacks often can be prevented through implementation refinement.

The notion of recognizability also facilitate a general framework to reason about off-line guessing attacks. The new framework is

- faithful (fits the common sense of guessing attacks),
- expressive (accounts for multiple guesses), and
- complete (captures all guessing attacks in a symbolic setting).

Moreover, this framework enables us to characterize the computational difficulty of guessing by making a clear distinction between explicit guesses and implicit guesses.

REFERENCES

[1] Dictionary and thesaurus - merriam-webster online. `www.merriam-webster.com`.

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

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

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

[5] Abadi, M., and Rogaway, P. Reconciling two views of cryptography (the computational soundness of formal encryption). *J. Cryptol. 20*, 3 (2007), 395–395.

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

[7] Armando, A., Basin, D. A., Bouallagui, M., Chevalier, Y., Compagna, L., Mödersheim, S., Rusinowitch, M., Turuani, M., Viganò, L., and Vigneron, L. The aviss security protocol analysis tool. In *CAV '02: Proceedings of the 14th International Conference on Computer Aided Verification* (London, UK, 2002), Springer-Verlag, pp. 349–353.

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

[9] Baskar, A., Ramanujam, R., and Suresh, S. P. Knowledge-based modelling of voting protocols. In *Proceedings of the 11th conference on Theoretical aspects of rationality and knowledge* (New York, NY, USA, 2007), TARK '07, ACM, pp. 62–71.

[10] Baudet, M. Deciding security of protocols against off-line guessing attacks. In *CCS '05: Proceedings of the 12th ACM conference on Computer and communications security* (New York, NY, USA, 2005), ACM, pp. 16–25.

[11] Birkhoff, G. On the structure of abstract algebras. *Mathematical Proceedings of the Cambridge Philosophical Society 31*, 04 (1935), 433–454.

[12] Blanchet, B. An efficient cryptographic protocol verifier based on prolog rules. In CSFW '01: Proceedings of the 14th IEEE workshop on Computer Security Foundations (Washington, DC, USA, 2001), IEEE Computer Society, p. 82.

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

[14] Blanchet, B., Abadi, M., and Fournet, C. Automated verification of selected equivalences for security protocols. Journal of Logic and Algebraic Programming 75, 1 (2008), 3 – 51. Algebraic Process Calculi. The First Twenty Five Years and Beyond. III.

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

[16] Bonneau, J. The science of guessing: Analyzing an anonymized corpus of 70 million passwords. In Security and Privacy (SP), 2012 IEEE Symposium on (may 2012), pp. 538 –552.

[17] Bonneau, J., Herley, C., van Oorschot, P., and Stajano, F. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In Security and Privacy (SP), 2012 IEEE Symposium on (may 2012), pp. 553 –567.

[18] Boreale, M., and Buscemi, M. G. A method for symbolic analysis of security protocols. Theoretical Computer Science 338, 1-3 (2005), 393 – 425.

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

[20] Burrows, M., Abadi, M., and Needham, R. A logic of authentication. ACM Trans. Comput. Syst. 8, 1 (1990), 18–36.

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

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

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

[24] Cervesato, I., Durgin, N. A., Lincoln, P. D., Mitchell, J. C., and Scedrov, A. A meta-notation for protocol analysis. In CSFW '99: Proceedings of the 12th IEEE workshop on Computer Security Foundations (Washington, DC, USA, 1999), IEEE Computer Society, p. 55.

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

[26] Chevalier, Y., and Vigneron, L. Automated unbounded verification of security protocols. In CAV '02: Proceedings of the 14th International Conference on Computer Aided Verification (London, UK, 2002), Springer-Verlag, pp. 324–337.

[27] Ciobâcă, c., Delaune, S., and Kremer, S. Computing knowledge in security protocols under convergent equational theories. In CADE-22 (Berlin, Heidelberg, 2009), Springer-Verlag, pp. 355–370.

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

[29] Cohen, M., and Dam, M. A complete axiomatization of knowledge and cryptography. In LICS 07 (Washington, DC, USA, 2007), IEEE Computer Society, pp. 77–88.

[30] Comon-Lundh, H., and Cortier, V. Computational soundness of observational equivalence. In CCS '08: Proceedings of the 15th ACM conference on Computer and communications security (New York, NY, USA, 2008), ACM, pp. 109–118.

[31] Comon-Lundh, H., Cortier, V., and Zălinescu, E. Deciding security properties for cryptographic protocols. application to key cycles. ACM Trans. Comput. Logic 11, 2 (2010), 1–42.

[32] Comon-Lundh, H., and Shmatikov, V. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In Proceedings of the 18th Annual IEEE Symposium on Logic in Computer Science (Washington, DC, USA, 2003), IEEE Computer Society, pp. 271–.

[33] Corin, R., Doumen, J., and Etalle, S. Analysing password protocol security against off-line dictionary attacks. Electron. Notes Theor. Comput. Sci. 121 (2005), 47–63.

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

[35] Corin, R., Malladi, S., Alves-Foss, J., and Etalle, S. Guess what? here is a new tool that finds some new guessing attacks. In IFIP WG 1.7 and ACM SIGPLAN Workshop on Issues in the Theory of Security (Warsaw, Poland, 2003), R. Gorrieri and R. Lucchi, Eds., Dpt. di Scienze dell'Informazione Universita di Bologna, pp. 62–71.

[36] Cortier, V., and Delaune, S. Deciding knowledge in security protocols for monoidal equational theories. In LPAR (2007), pp. 196–210.

[37] Cortier, V., and Delaune, S. A method for proving observational equivalence. In CSF '09: Proceedings of the 2009 22nd IEEE Computer Security Foundations Symposium (Washington, DC, USA, 2009), IEEE Computer Society, pp. 266–276.

[38] Cortier, V., Delaune, S., and Lafourcade, P. A survey of algebraic properties used in cryptographic protocols. J. Comput. Secur. 14, 1 (2006), 1–43.

[39] Cox, P. T., and Pietrzykowski, T. Causes for events: their computation and applications. In Proc. of the 8th international conference on Automated deduction (New York, NY, USA, 1986), Springer-Verlag New York, Inc., pp. 608–621.

[40] Cremers, C. J. 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 (New York, NY, USA, 2008), ACM, pp. 119–128.

[41] Dechesne, F., Mousavi, M., and Orzan, S. Operational and epistemic approaches to protocol analysis: bridging the gap. In Proceedings of the 14th international conference on Logic for programming, artificial intelligence and reasoning (Berlin, Heidelberg, 2007), LPAR'07, Springer-Verlag, pp. 226–241.

[42] Delaune, S. Easy intruder deduction problems with homomorphisms. Inf. Process. Lett. 97, 6 (2006), 213–218.

[43] Delaune, S., and Jacquemard, F. A decision procedure for the verification of security protocols with explicit destructors. In CCS '04: Proceedings of the 11th ACM conference on Computer and communications security (New York, NY, USA, 2004), ACM, pp. 278–287.

[44] Delaune, S., and Jacquemard, F. A theory of dictionary attacks and its complexity. In CSFW '04: Proceedings of the 17th IEEE workshop on Computer Security Foundations (Washington, DC, USA, 2004), IEEE Computer Society, p. 2.

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

[46] Dershowitz, N., and Plaisted, D. A. Rewriting. In Handbook of Automated Reasoning. MIT Press, 2001, pp. 535–610.

[47] Ding, Y., and Horster, P. Undetectable on-line password guessing attacks. SIGOPS Oper. Syst. Rev. 29, 4 (1995), 77–86.

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

[49] Drielsma, P. H., Modersheim, S., and Vigano, L. A formalization of off-line guessing for security protocol analysis. In Logic for Programming, Artificial Intelligence, and Reasoning, F. Baader and A. Voronkov, Eds., Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2005, pp. 363–379.

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

[51] Durgin, N., Lincoln, P., Mitchell, J., and Scedrov, A. Multiset rewriting and the complexity of bounded security protocols. J. Comput. Secur. 12, 2 (2004), 247–311.

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

[53] Fábrega, F. J. T. Strand spaces: proving security protocols correct. J. Comput. Secur. 7, 2-3 (1999), 191–230.

[54] Fagin, R., Halpern, J. Y., Moses, Y., and Vardi, M. Y. Reasoning About Knowledge. MIT Press Books. The MIT Press, December 2003.

[55] Gong, L. Optimal authentication protocols resistant to password guessing attacks. In CSFW '95: Proceedings of the 8th IEEE workshop on Computer Security Foundations (Washington, DC, USA, 1995), IEEE Computer Society, p. 24.

[56] Gong, L., Lomas, M., Needham, R., and Saltzer, J. Protecting poorly chosen secrets from guessing attacks. Selected Areas in Communications, IEEE Journal on 11, 5 (jun. 1993), 648 –656.

[57] Gong, L., Needham, R., and Yahalom, R. Reasoning about belief in cryptographic protocols. In Proc. of IEEE Symposium on Security and Privacy (1990), pp. 238–248.

[58] Gordon, A. D., and Jeffrey, A. Types and effects for asymmetric cryptographic protocols. J. Comput. Secur. 12 (May 2004), 435–483.

[59] Groza, B., and Minea, M. A calculus to detect guessing attacks. In ISC '09: Proceedings of the 12th International Conference on Information Security (Berlin, Heidelberg, 2009), Springer-Verlag, pp. 59–67.

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

[61] Hahnle, R. Tableaux and related methods. In Handbook of Automated Reasoning. MIT Press, 2001, pp. 103–177.

[62] Halevi, S., and Krawczyk, H. Public-key cryptography and password protocols. ACM Trans. Inf. Syst. Secur. 2, 3 (1999), 230–268.

[63] Halpern, J., Moses, Y., and Vardi, M. Algorithmic knowledge. In Proc. of 5th conference on Theoretical Aspects of Reasoning about Knowledge (1994), pp. 255–266.

[64] Halpern, J. Y., and Pucella, R. Modeling adversaries in a logic for security protocol analysis. CoRR abs/cs/0607146 (2006).

[65] Heather, J., Lowe, G., and Schneider, S. How to prevent type flaw attacks on security protocols. J. Comput. Secur. 11, 2 (2003), 217–244.

[66] Hunter, A., and Delgrande, J. P. Belief change and cryptographic protocol verification. In Proceedings of the 22nd national conference on Artificial intelligence - Volume 1 (2007), AAAI Press, pp. 427–433.

[67] Katz, J., Ostrovsky, R., and Yung, M. Efficient and secure authenticated key exchange using weak passwords. J. ACM 57, 1 (Nov. 2009), 3:1–3:39.

[68] Kripke, S. Naming and Necessity. Harvard University Press, 1980.

[69] Kripke, S. A. Semantical analysis of modal logic i normal modal propositional calculi. Mathematical Logic Quarterly 9, 5-6 (1963), 67–96.

[70] Lafourcade, P., Lugiez, D., and Treinen, R. Intruder deduction for the equational theory of abelian groups with distributive encryption. Inf. Comput. 205, 4 (2007), 581–623.

[71] Landauer, R. Irreversibility and heat generation in the computing process. IBM Journal of Research and Development 44, 1.2 (jan. 2000), 261 –269.

[72] Li, Z., and Wang, W. Rethinking about type-flaw attacks. In GLOBECOM 2010 (2010), pp. 1 –5.

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

[74] Li, Z., and Wang, W. Rethinking about guessing attacks. In Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security (New York, NY, USA, 2011), ASIACCS '11, ACM, pp. 316–325.

[75] Li, Z., and Wang, W. Towards the attacker's view of protocol narrations (or, how to compile security protocols). In Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security (New York, NY, USA, 2012), ASIACCS '12, ACM.

[76] Lomas, T., Gong, L., Saltzer, J., and Needhamn, R. Reducing risks from poorly chosen keys. SIGOPS Oper. Syst. Rev. 23, 5 (1989), 14–18.

[77] Lomuscio, A., and Woźna, B. A complete and decidable security-specialised logic and its application to the tesla protocol. In AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems (New York, NY, USA, 2006), ACM, pp. 145–152.

[78] Long, B., Fidge, C., and Carrington, D. Cross-layer verification of type flaw attacks on security protocols. In Thirtieth Australasian Computer Science Conference (ACSC2007) (Ballarat Australia, 2007), G. Dobbie, Ed., CRPIT, ACS, pp. 171–180.

[79] Long, B. W. Formal verification of type flaw attacks in security protocols. In APSEC '03: Proceedings of the Tenth Asia-Pacific Software Engineering Conference Software Engineering Conference (Washington, DC, USA, 2003), IEEE Computer Society, p. 415.

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

[81] Lowe, G. Breaking and fixing the needham-schroeder public-key protocol using fdr. In Proceedings of the Second International Workshop on Tools and Algorithms for Construction and Analysis of Systems (London, UK, UK, 1996), TACAs '96, Springer-Verlag, pp. 147–166.

[82] Lowe, G. Breaking and fixing the needham-schroeder public-key protocol using fdr. In TACAs '96 (1996), pp. 147–166.

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

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

[85] Lowe, G. Analysing protocols subject to guessing attacks. J. Comput. Secur. 12, 1 (2004), 83–97.

[86] Meadows, C. Open issues in formal methods for cryptographic protocol analysis. In Proceedings of the International Workshop on Information Assurance in Computer Networks: Methods, Models, and Architectures for Network Security (London, UK, UK, 2001), MMM-ACNS '01, Springer-Verlag, pp. 21–.

[87] Meadows, C. Identifying potential type confusion in authenticated messages. In In Proceedings of Foundations of Computer Security 02 (2002), pp. 75–84.

[88] Meadows, C. Formal methods for cryptographic protocol analysis: emerging issues and trends. Selected Areas in Communications, IEEE Journal on 21, 1 (Jan 2003), 44–54.

[89] Meadows, C. A procedure for verifying security against type confusion attacks. In CSFW 03 (2003), IEEE Computer Society Press, pp. 62–72.

[90] Meadows, C., and Meadows, C. Formal verification of cryptographic protocols: A survey. In Advances in Cryptology, ASIACRYPT'94, J. Pieprzyk and R. Safavi-Naini, Eds., Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 1995, pp. 133–150.

[91] Meyden, R. v. d., and Su, K. Symbolic model checking the knowledge of the dining cryptographers. In Proceedings of the 17th IEEE workshop on Computer Security Foundations (Washington, DC, USA, 2004), IEEE Computer Society, pp. 280–.

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

[93] Millen, J., and Shmatikov, V. Symbolic protocol analysis with an abelian group operator or diffie–hellman exponentiation. J. Comput. Secur. 13, 4 (2005), 695–695.

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

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

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

[97] Peterson, J. Dynamic typing in haskell. Tech. Rep. YALEU/DCS/RR-1022, Department of Computer Science, Yale University, 1993.

[98] Pucella, R. Deductive algorithmic knowledge. J. Log. and Comput. 16 (April 2006), 287–309.

[99] Ramanujam, R., and Suresh, S. P. Deciding knowledge properties of security protocols. In Proceedings of the 10th conference on Theoretical aspects of rationality and knowledge (Singapore, Singapore, 2005), TARK '05, National University of Singapore, pp. 219–235.

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

[101] Schneider, S. Security properties and csp. In SP '96: Proceedings of the 1996 IEEE Symposium on Security and Privacy (Washington, DC, USA, 1996), IEEE Computer Society, p. 174.

[102] Smith, R. E. Authentication: From Passwords to Public Keys. October 2001.

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

[104] Stubblebine, S. G., and Wright, R. N. An authentication logic with formal semantics supporting synchronization, revocation, and recency. IEEE Trans. Softw. Eng. 28, 3 (2002), 256–285.

[105] Syverson, P. Formal semantics for logics of cryptographic protocols. In Computer Security Foundations Workshop III, 1990. Proceedings (June 1990), pp. 32 –41.