Doctoral Dissertation

Improving Formal Analysis Method with Reasoning for Cryptographic Protocols

暗号プロトコルのための前向き推論を用いた形式分析手法の改善

Jingchen Yan

Graduate School of Science and Engineering,
Saitama University

Supervisor: Professor Yuichi Goto

March 2019

Abstract

Cryptographic protocols are protocols which perform some security-related functions using cryptography. Flaws of cryptographic protocols will bring serious security problems to the cyberspace application, and even cause the immeasurable loss. Therefore, security analysis of the cryptographic protocols become an indispensable process.

Formal analysis method is used to security analysis method for cryptographic protocols. Until now, theorem proving method and model checking method are widely used for formal analysis of cryptographic protocols. These methods are proving methods because analysts should enumerate the concrete security specifications in advance, then prove or check whether the target cryptographic protocol satisfies the enumerated security specifications or not, thereby verifying whether flaws exist or not. Therefore, when using the formal analysis method with proving to verify the security of a cryptographic protocol, all the concrete security specifications must be enumerated. However, it is generally difficult for analysts to enumerate the security specifications completely.

As an alternative way, a concept of formal analysis method with reasoning has been proposed. Analysts do not need to enumerate all the security specifications in advance but take the behaviors explicitly and implicitly included in the specifications of cryptographic protocols as premises to perform forward reasoning. By forward reasoning, formulas related to flaws can be deduced and it is possible to deduce the formulas that point to some unknown flaws in principle. Wagatsuma et. al proposed the concrete tasks of formal analysis method with reasoning for key exchange protocols. In the method, analysts formalize the participant 's behaviors and an intruder 's behaviors to perform forward reasoning, and then analysts analyze the deduced formulas whether successful attacks exist. However, there are still some limitations and problems in the method. First, it cannot analyze the cryptographic protocols except key exchange protocols. Second, there are no clear flaw analysis criteria to accurately analyze whether the deduced formulas are related to the flaws. Third, due to the limitations of the participants' behaviors, some flaws cannot be detected. Fourth, many tasks in formal analysis need to perform manually that lead to the time-consuming and error-prone problems.

This thesis proposes the improving formal analysis method with reasoning for cryptographic protocols and the supporting environment for formal analysis to solve the above limitations and problems. First, we extended the formal analysis method with reasoning to apply to various cryptographic protocols. Second, we proposed the fine-grained flaw analysis criteria to analyze the deduced formulas. Third, we extended the formalization tasks and forward reasoning tasks to detect more types of flaws. Fourth, we proposed the supporting environment for formal analysis. This thesis describes the findings and results obtained through our research on demonstrating the effectiveness of the extended method and the support environment and is composed of 8 Chapters.

We described the extended formal analysis method with reasoning which can be applied to various cryptographic protocols in chapter 3. As the first step of expansion, we compared 19 representative cryptographic protocols based on participants 'number and behaviors and summarized five features. Then, we extended the participants 'behaviors

corresponding to the summarized features. We also proposed a specific procedure to decided how falsified data that an intruder would send. After extending, we performed case studies to verify the extended method can be used to detect various cryptographic protocols. By succeeding detecting the known flaws of secret splitting protocols, it can be said that the extended method is effective to apply to various cryptographic protocols.

We proposed the flaw analysis criteria have been proposed in chapter 4. We defined what is a flaw in a cryptographic protocol and considered the security properties of cryptographic protocols can be used as criteria for testing the security of cryptographic protocols. We have organized six fine-grained security properties of confidentiality, authentication, fairness, non-repudiation, anonymity, and atomicity, and used them as flaw analysis criteria to analyze flaws in cryptographic protocols. We also verified the flaw analysis criteria are capable by analyzing Needham Schroeder protocol and anonymous atomic transaction protocols.

To detect more types of cryptographic protocol flaws, we proposed a new method to perform forward reasoning to test the flaws related to non-repudiation and fairness caused by participants' deception in Chapter 5. We also took the ISI protocol and CMP1 protocol as cases to perform security analysis. By the result of analyzing, it is proved that the improved formal analysis method with reasoning is effective to detect the flaws related to non-repudiation and fairness.

To resolve the time-consuming and error-prone problems, we explained the first supporting environment for formal analysis of cryptographic protocols in chapter 6. The supporting environment integrates various supporting tools and it can support analysts to perform formal analysis of cryptographic protocols through the whole processes.

In chapter 7, we discussed soundness of the method proposed in this research and the limitations of the extended method.

Finally, Chapter 8 summarizes the results obtained in this study and shows the future works.

Acknowledgments

Before the thesis, I remembered a sentence. The most valuable thing you take away from here is not a paper, but the ability to analyze and think, the experience of discovering the truth, and the mind of a scientist.

During the two years of my Ph.D., I would like to express my special thanks to my thesis supervisor, Professor Yuichi Goto for his invaluable support, enthusiastic guidance and understanding on all aspects of academic life to help me accomplish my thesis and pursue my doctoral degree. From the structure of this thesis to the text retouching, every aspect has the careful guidance of Professor Goto.

I am very grateful to Professor Jingde Cheng, my master's supervisor, for his care and support during the past three years. Thanks to Professor Cheng for taking me into the door of scientific research and his teachings and dedications, so that I can go through the hard moments and move towards the truth.

I am also grateful to my dissertation committee: Professor Norihiko Yoshida, Professor Noriaki Yoshiura and Professor Jun Ohkubo for their support, valuable feedback, and insightful ideas to this research.

I would like to show my thanks to the former Assistant Professor Hongbiao Gao. Thanks for providing me with too much help, and discussing with me after going back to China. I am also grateful to other AISE lab members, Da Bao, Yuan Zhou, Sho Ishibashi, Yating Wang to give me advice and help me during my Ph.D. life.

At last, I want to express my special thanks to my husband, Doctor Wenjing Zhang, my parents, and my friends Tianjiao Liang and Niankai Zhang. Thanks for their great support and encouragement in my life. They are my strong backing so that I can go forward without hesitation.

Sincerely thank you all, for letting me become not vulnerable, not silent, not compromise, not retreat, not panic, not despair, not arrogance and not surrender, and for letting me fly all the way.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

In the highly informative society, cryptographic protocols are necessary techniques to ensure the security and fairness of many applications in cyberspace. Cryptographic protocols are protocols which perform some security-related functions using cryptography. Cryptographic protocols are protocols which perform some security-related functions using cryptography. Flaws of cryptographic protocols will bring serious security problems to the cyberspace application, and even cause the immeasurable loss. Therefore, security analysis of the cryptographic protocols become an indispensable process.

Formal analysis method is used for security analysis of cryptographic protocols[23][41] [43]. Until now, as formal analysis methods with proving, theorem proving method and the model checking method [6][8][21][50] are widely used for formal analysis of cryptographic protocols. These methods can be regarded as proving methods because analysts should enumerate the security properties that a protocol should satisfy and accurately describe these properties as formulas in advance, then prove or check whether the target cryptographic protocol satisfies the formulas or not, thereby verifying whether flaws exist or not. Theoretical limitation of such proving methods is that analysts should enumerate all formulas accurately describe the security properties before they start to verify target cryptographic protocols with the methods and the methods only can verify the security through the enumerated formulas. If the enumeration is not enough, some flaws cannot be detected. However, it is generally difficult for analysts to enumerate the formulas completely.

As an alternative way, a concept of formal analysis method with reasoning has been proposed by Cheng [19]. In the method, it is not necessary to enumerate all the formulas that accurately describe the properties in advance but takes the behaviors of participants and the behaviors of an intruder as premises to perform forward reasoning. By forward reasoning, formulas related to flaws can be deduced and it is possible to deduce some formulas that point to some unknown flaws in principle. Wagatsuma et. al [55] proposed the concrete procedures in formal analysis method with reasoning to detect key exchange protocols. By using this method, flaws of Otway-Rees protocol [46] can be detected. However, there are still some limitations and problems in the method. First, it cannot be used to detect flaws of various cryptographic protocols except key exchange

protocols. Second, there are no clear flaw analysis criteria to accurately analyze whether the deduced formulas are related to flaws of the target protocols. Third, due to the limitations of the behaviors, some flaws cannot be detected. Fourth, many tasks in the method need to perform manually that lead to time-consuming and error-prone problems.

1.2 Purpose and Objectives

Purpose of the research is to improve the formal analysis method with reasoning for key exchange protocols by solving the above limitations and problems. At first, we extended the formal analysis method with reasoning for key exchange protocols to make it possible to apply to various cryptographic protocols. Then we defined what is a flaw in a cryptographic protocol and propose the security properties of cryptographic protocols can be used as criteria for testing the security of cryptographic protocols. After that, to detect more kinds of flaws, we proposed a new method to perform forward reasoning to find out flaws related to non-repudiation and fairness caused by participants' deception. At last, we proposed the first supporting environment for formal analysis of cryptographic protocols.

1.3 Structure of This Thesis

The rest of this thesis is organized as follows. Chapter 2 explains formal analysis of cryptographic protocols. Chapter 3 shows the improvement of applying formal analysis method with reasoning to various cryptographic protocols. Chapter 4 describes the improvement of flaw analysis criteria. Chapter 5 presents the improvement of detecting more flaws of cryptographic protocols. Chapter 6 proposes the first supporting environment for formal analysis. Chapter 7 discuss the problems of the extended method. Finally, contributions and future works are given in Chapter 8.

Chapter 2

Formal Analysis of Cryptographic protocols

2.1 Basic Notions of Cryptographic Protocols

2.1.1 Cryptographic Protocols and Classification

A protocol is a series of steps, involving two or more entities, designed to accomplish a task. A "series of steps" means that the protocol has a sequence, from start to finish. Every step must be executed in turn, and no step can be taken before the previous step is finished. "Involving two or more entities" means that at least two people are required to complete the protocol [3]. A cryptographic algorithm, also called a cipher, is the mathematical function used for encryption and decryption. A cryptographic protocol is a protocol that uses cryptography.

A cryptographic protocol is a protocol which performs a security-related function using cryptography [37]. Security-related function means prevention of or protection against (a) access to information by an unauthorized entity or (b) the intentional but unauthorized destruction or alteration of that information [15].

Until now, no one has classified the cryptographic protocol in detail. In fact, strict classification of cryptographic protocols is very difficult. From different perspectives, there are different classification methods. For example, according to the ISO reference model, the cryptographic protocol can be divided into a high-level protocol and a low-level protocol; according to the encryption algorithms to be used, cryptographic protocols can be divided into public key protocol, symmetric key protocol, and hybrid protocol. In this thesis, we divide the cryptographic protocols into the following four types according to the function of the cryptographic protocol.

1. Key exchange protocol

   In order to achieve communication between entities, the key exchange protocol shares a secret session key between two or more entities through a series of message interactions [12].

2. Authenticated protocol

The authenticated protocol realizes the authentication of the entities' identity or the data source through the interaction of the message, preventing the intruder from impersonating or tampering with the data attack [22].

3. Authenticated key exchange protocol

   The authenticated key exchange protocol is to implement entity authentication or data source authentication while sharing a secret session key [30].

4. E-commence protocol

   E-commerce protocol is a cryptographic protocol to ensure normal, reliable, and secure transactions between participants (customers, banks, merchants, and other entities) in e-commerce activities [37].

2.1.2 Composition of Cryptographic Protocols

According to the definition of Cryptographic protocols, Ishibashi et.al [34] proposed the primitive constituent elements of cryptographic protocols. Here, we consider that a cryptographic protocol consists of participants and the message they exchange.

Entity is an independent unit which performs one or more operations in a cryptographic protocol. There are two types of entity, *Agent* and $Sever$. *Agent* is an independent unit which actively performs each operation. $Sever$ is an independent system which passively returns an answer to the request. *Participant* is an authorized entity who executes the cryptographic protocols. *Participant* includes a variety of roles, such as sponsors and responders in authentication protocols; trusted servers in arbitrated protocols; prover and verifier in zero-knowledge protocols; merchants, banks and customers in E-commerce protocols. *Intruder* is an unauthorized entity who participates in or interfere with the execution of cryptographic protocols.

In a cryptographic protocol, $Message$ consists of *Data* and *Item*. *Data* is a primitive and independent unit. *Item* is a package that includes some *Data* and *Item*. $SecretData$ is the independent unit represent secret information. $SecretItem$ is the *Item* which includes secret information. $SecretData$ and $SecretItem$ are secret information. In general, a cryptographic protocol is described as the following format.

1. $A \rightarrow S : A, B, N_a$
2. $S \rightarrow A : \{N_a, B, SK, \{SK, A\}_{K_b}\}_{K_a}$

In the protocol, A, B, N_a, SK, these four are all *Data*, where SK is $SecretData$. $\{SK, A\}_{K_b}$, $\{N_a, B, SK, \{SK, A\}_{K_b}\}_{K_a}$, both are *Item*. SK, $\{SK, A\}_{K_b}$ are the $SecretItem$. The sentence "A, B, N_a" and "$\{N_a, B, SK, \{SK, A\}_{K_b}\}_{K_a}$" are both $Message$.

Cryptographic protocols deal with several kinds of keys. Long-term key (it is also called private key) and public key are the asymmetric key pairs that are used by public key algorithms. Long-term key is with possible long-term implications. It is deliberately stored somewhere, either on a computer disk, flash memory, or even printed on paper. The key is intended to be used at multiple points in time. Symmetric key and session key are used by symmetric algorithms. Symmetric key is generally agreed by the participants of a protocol before communication and can be reused for a long time. Session

key is a single-use symmetric key used for encrypting all messages in one communication session. It is not intentionally stored and is not re-creatable. Session keys are used to encrypt the message only as long as the conversation of cryptographic protocols.

Fresh identifier is the unique identifier generated for the execution of a protocol. It can be a random number, a timestamp, a session key, or a component that generates a new session key. If an identifier or a message is generated by a participant specifically for this protocol or is transformed with this specially generated content in a one-way transmission, and the composite message involves the cryptographic operation corresponding to the responder, then participant can confirm that the identifier or message is fresh.

2.1.3 Security of Cryptographic Protocols

When analyzing the security of cryptographic protocols, a common idea is to use various possible attacks to test the cryptographic protocol's security. There are usually three targets for attacks, one is the cryptographic algorithm used by the cryptographic protocol, the other is the cryptographic technique used in the algorithm and protocol, and the third is the cryptographic protocol itself. In this thesis, we mainly study the attack on the protocol itself and assume that the cryptographic algorithms and cryptographic techniques used in the protocol are all secure.

Attacks on the cryptographic protocol itself can be divided into active attacks and passive attacks. Passive attack refers to an entity outside the protocol that eavesdrops on part or the whole process of protocol execution. We call the outside entity an intruder. The intruder's eavesdropping on the protocol does not strictly enforce the implementation of that protocol. What an intruder can do is to observe the information of the cryptographic protocol and try to get some information from the participants. The characteristics of passive attacks are difficult to detect. Active attacks are more dangerous for cryptographic protocols. In this type of attacks, an intruder attempts to change certain messages in the execution of the protocol, destroy the system, or gain unauthorized access to the resource. They may introduce new messages in the protocol, delete messages, replace messages, and resend old ones.

In the communication process of a cryptographic protocol, an attack is a sequence of actions by participants and an intruder. A successful attack is that an intruder can do the last action of the attack. In this thesis, we call the last action of an attack "a fatal action of an attack". If successful attacks occurred in the communication process of the cryptographic protocol, we say that the cryptographic protocol has flaws. A behavior is a set of rules among events and/or actions in the communication processes of cryptographic protocols.

About the attacks, Dolev and Yao proposed the Dolev-Yao intruder model [29]. They proposed to separate the cryptographic protocol itself from the cryptographic algorithm specifically adopted by the cryptographic protocol, and analyze the correctness, security, and redundancy of the security protocol itself on the assumption of a perfect cryptosystem. They also built an intruder model that accurately portrays the intruder's behaviors. It defines that

1. The intruder can obtain any message passing through the network.

2. The intruder is a legitimate user of the network, and thus, in particular, can initiate a conversation with any other user.

3. The intruder will have the opportunity to be a receiver to any user A. (More generally, we allow the possibility that any user B may become a receiver to any other user A.)

2.2 Formal Analysis Method with Proving

Model checking and theorem proving are formal analysis method with proving.

In model checking method, a cryptographic protocol is described by the state transition system and the security properties that the protocols should satisfied are described as temporal logic formulas, then analysts verify whether the target cryptographic protocol satisfies the security specifications [6][8][21]. Temporal logic can assert how the behavior of the system evolves over time [7]. For example, in NSSK protocol, one of the security properties is "When A and B complete the protocol, both participants correctly record the identity of the other", and the temporal logic formula is represented by

$(A.success, B.success) imply A.myNonece == B.yourNonce \wedge A.yourNonce == B.myNonce$

Also its supporting tool such as Scyther [24], ProVerif [9] have been proposed.

Theorem proving is also based on proving. A cryptographic protocol is constructed into as formal theorems and the security properties are formalized as model theorem. Then analysts should prove whether the formal theorems hold or not by using inference rules and model theorem [49]. CafeOBJ [31] and Isabelle [50] are proposed and applied as typical supporting tools.

However, there is a limitation of formal analysis method with proving, that is, analysts should enumerate all formulas accurately describe the security properties before they start to verify target cryptographic protocols with the methods and the methods only can verify the security through the enumerated formulas. If the enumeration is not enough or incorrect, some flaws cannot be detected. However, it is generally difficult for analysts to enumerate the formulas completely.

2.3 Formal Analysis Method with Reasoning

As an alternative way, a method of formal analysis method with reasoning for cryptographic protocols has been proposed by Cheng [19]. In this method, analysts take formalized explicitly and implicitly behaviors included in the specifications of a protocol as premises to perform forward reasoning. By forward reasoning, formulas related to flaws can be deduced and it is possible to deduce the formulas that point to some unknown flaws in principle. Wagatsuma et. al [55] proposed the concrete procedures of formal analysis method with reasoning for key exchange protocols. In the method, analysts formalize the participant 's behaviors and an intruder 's behaviors to perform forward reasoning, and Then analyze the results of forward reasoning whether fatal actions exist. If fatal actions exist, attacks succeed in the target cryptographic protocol. By forward reasoning, all the possibility of the formulas that represent fatal actions of

attacks can be deduced in principle. In other words, we can detect all the successful attacks so that we can find out all the flaws of the protocol in principle.

In the method, strong relevant logics [17][18] are appropriate logic systems for underlying forward reasoning because conclusions not related to premises cannot be deduced. Furthermore, forward reasoning engine FreeEnCal [20] that can automatically perform forward reasoning has been proposed and developed. A support tool for spiral model of cryptographic protocol design also proposed [56].

However, there are some limitations and problems in Wagatsuma's method. First, the target of this method is limited to only key exchange protocols. Second, there are no clear flaw analysis criteria to accurately analyze whether the deduced formulas are related to flaws of the target protocol. Third, due to the limitations of the behaviors, some flaws cannot be detected. Fourth, many tasks in the method need to perform manually that lead to time-consuming and error-prone problems. Therefore, the method should be improved in the above three aspects, and an automated support environment is needed.

Chapter 3

Improvement of Applying to Various Cryptographic Protocols

To apply the formal analysis method with reasoning to various cryptographic protocols, we compared 19 representative cryptographic protocols based on participants' number and behaviors and summarized five features of these protocols. Then, we extended the participants ' behaviors by adding new rules corresponding to the summarized features. We also proposed a specific procedure to decided how falsified data that an intruder would send. After extending, we performed case studies to verify whether the extended method can be used to detect flaws of various cryptographic protocols.

3.1 Features of Cryptographic Protocols

We compiled 19 representative cryptographic protocols [45], compared the differences based on participants' number and behaviors and summarized five features of these protocols. Table 3.1 shows the features that different protocols have.

F1: There are three or more participants in a cryptographic protocol.

F2: There is one or more trusted server in a cryptographic protocol.

F3: Participants send data selectively to another in a cryptographic protocol.

F4: One participant sends data to multiple participants in a cryptographic protocol.

F5: Participants get another data by calculation from collected data in a cryptographic protocol.

Kind of Protocol	F1	F2	F3	F4	F5
Key Exchange	○	○			○
Authentication	○	○			○
Secret Splitting	○	○		○	○
Timestamp		○			
Subliminal Channel		○			○
Undeniable Digital Signatures					○
Bit Commitment				○	
Coin Flips			○		
Mental Poker	○		○		
Anonymous Key Distribution		○	○		
Key Escrow	○	○		○	
Zero Knowledge Proofs			○		
Blind Signatures					○
Oblivious Transfer			○		
Simultaneous Contact Signing	○				
Digital Certified Mail			○	○	○
Secure Elections					
Secure Multiparty Computation	○				
Digital Cash	○				

Table 3.1: Features of cryptographic protocols

3.2 Formalization

3.2.1 Overview of Formalization

Analysts formulate "behaviors of participants", "behaviors of an intruder" and "common behaviors among participants and an intruder" as targets of formalization. "Behaviors of participants" means a set of rules of participants." "Behaviors of an intruder" means a set of rules what actions an intruder performs. "Common behaviors among participants and an intruder" represents rules except the behavior about sending and receiving data by participants and an intruder.

In this method, analysts formalize the above behavior rules based on first order predicate strong relevant logics [17][18]. In order to be applied to various cryptographic protocols, we defined following predicates, functions, and individual constants that represents participants' behavior or data in cryptographic protocols. We defined following predicates, functions and individual constants.

Predicates

- $Parti(p)$: p is a participant of a protocol.
- $Eq(x_1, x_2)$: x_1 and x_2 are equal.
- $Get(p, x)$: p gets x.

- $Recv(p, x)$: p receives x.
- $Send(p_1, p_2, x)$: p_1 sends x to p_2.
- $Start(p_1, p_2)$: p_1 and p_2 start a communication process.

Functions

- $data(x_1, \cdots, x_n)$ ($n \in \mathbb{N}$): A data set that consists of sent and received $x_1, \ldots,$ and x_n.
- $enc(k, x_1, \cdots, x_n)$: A data set that consists of encrypted $x_1, \cdots,$ and x_n by k.
- $id(p)$: Identifier of p.
- $nonce(p)$: Nonce of p.
- $old(x)$: Old data of x.
- $pk(p)$: Public key of p.
- $plus(x)$: Incremented data of x.
- $sig(p, x_1, \cdots, x_n)$: A data set that consists of $x_1, \cdots,$ and x_n with p's signature.
- $symk(p_1, p_2)$: Symmetric key of p_1 and p_2.
- $tstamp(p)$: Timestamp of p.

Individual constants

- $a, b, \ldots, h, a_1, \ldots, a_n$ ($n \in \mathbb{N}$): Persons
- i: An intruder.
- $s_1, \ldots, s_n$ ($n \in \mathbb{N}$): Trusted servers.

In addition, there are uniquely defined functions and constants that are assigned to uniquely defined data in a protocol.

In the proposed method, we assumed following premises about participants' and an intruder's behavior. At first, if received data that participants have is falsified, the participants can recognize falsification. Second, an intruder does not forge sending data if participants can recognize falsification. Finally, participants know what kinds of data are derived to them as the next data.

In order to extend the formalization procedure, we add a case that there are three or more participants to corresponding to F1, a case that there are multiple trusted servers to corresponding to F2, and a case that a participant selectively sends data to corresponding to F3 in section 3.2.2. we add a case that a participant sends data to multiple participants to corresponding to F4 in section 3.2.5. we add a task of generating logical formulas that participants get new data by calculating got multiple data to corresponding to F4 in section 3.2.4.

In the formalization progress, three sets of formulas are generated, Normal Path (NP_n^m), Irregular Path (IP_n^m), and Common Input (CO). NP_n^m represents that a participant correctly sends data in step n. n represents the number of steps. m represents the number of branch in case that participants send data selectively to another in step n. IP_n^m represents that an intruder sends data in step n. m represents the number of branch by falsification. CO is a set of logical formulas that represents implicit participants' and an intruder's behavior.

3.2.2 Behaviors of Participants

1. Represent participants' behavior in each step of the protocol by formulas.

   a. As step 1 of the protocol,

   $$Start(p_1, p_2) \Rightarrow Send(p_1, p_2, data(x_1, \ldots, x_n)) \tag{3.1}$$

   means "if a communication process starts with p_1 and p_2, p_1 sends data $x_1, \ldots, x_n$ to p_2".

   b. As step 2 of the protocol,

   $$Recv(p_1, data(x_1, \ldots, x_m)) \Rightarrow (Parti(p_1) \Rightarrow Send(p_1, p_2, data(x_1, \ldots, x_n))) \tag{3.2}$$

   means "if a participant p_1 receives data, p_1 sends next data to p_2." But p_i, x_i are individual variables, and n, m are the number of sent or received data. If a participant selectively sends data in a corresponding step, logical formulas about corresponding step are generated in section 3.2.5.

2. Replace individual variables p_1 and p_2 of formulas (3.1) and (3.2) with s_n or p_i respectively in the previous task. For example, if sender of corresponding step is a trusted server $S_1, S_2, \ldots$, or S_n, individual variable p_1 is replaced with $s_1, s_2, \ldots$, or s_n, respectively.

3. Replace individual variables $x_1, \cdots, x_n$ of formulas (3.1) and (3.2) with terms according to following rules corresponding step of the specification.

   a. If sent data Y_i is not encrypted, substitute a function $f(p_i)$ or $f(s_n)$ respectively or an individual variable that is uniquely defined.

   b. If Y_i is incremented data, substitute $plus(x_i')$. x_i' is replaced as well as previous task 3-a.

   c. If Y_i is encrypted data, substitute $enc(k, x_1', \cdots, x_n')$ and replace k depending on key types such as public key or symmetric key.

4. In part of formulas $A_1 \Rightarrow A_2$ (A_1, A_2 is formulas), if a variable is included only in A_1 or A_2, define an individual constant and replace the variable into the constant.

5. Add quantifier $\forall$ corresponded to individual variables k, x_i, and p_i in those formulas.

6. Add generated logical formulas that represent the behavior of step i in NP_i^1.

7. Add a formula $Start(p_1, p_2)$ in NP_1^1 with substituting an individual constant of participants or an intruder to p_1 and p_2.

3.2.3 Behaviors of An Intruder

The behavior of an intruder in the proposed method is based on Dolev-Yao model [29]. Therefore, formalization of tasks are as follows.

1. Enumerate all functions $data(x_1, \ldots, x_n)$ in predicate $Send$ that is generated based on formulas (4.1) and (4.2). Then, Generate $Start(p_1, p_2) \Rightarrow Get(i, T_m)$ where T_i is enumerated $data(x_1, \ldots, x_n)$ and $1 \leq i \leq n$. m is number of enumerated $data(x_1, \ldots, x_n)$.

2. If T_m includes function *nonce* or *tstamp* or any individual constants, those terms y are replaced into $old(y)$.

3. Add generated formulas and a formula $\forall p_1 \forall p_2 \forall x(Send(p_1, p_2, x) \Rightarrow Get(i, x))$ which represents "if a participant sends x, an intruder gets it (by eavesdropping)"in CO.

3.2.4 Common Behavior among Participants and An Intruder

Analysts add following formulas in CO depending on number of participants and used key.

1. Add $\forall p((Get(p, getdata_1) \wedge \cdots \wedge Get(p, getdata_n) \Rightarrow Get(p, getdata')))$ represents if p gets multiple data, it gets another data.

2. Add the formula that means if p gets data encrypted by p's symmetric key or public key, p_1 gets original data. $\forall p_1 \forall p_2 \forall x_1 \ldots \forall x_n (Get(p_1, enc(symk(p_1, p_2), x_1, \cdots, x_n)) \Rightarrow Get(p_1, data(x_1, \cdots, x_n)))$, and
   $\forall p \forall x_1 \ldots \forall x_n (Get(p, enc(pk(p), x_1, \cdots, x_n)) \Rightarrow Get(p, data(x_1, \cdots, x_n)))$

3. $\forall p \forall x (Recv(p, x) \Rightarrow Get(p, x))$ represents if p receives x, p gets it.

4. $\forall p_1 \forall p_2 (Eq(symk(p_1, p_2), symk(p_2, p_1)))$ represents $symk(p_1, p_2)$ and $symk(p_2, p_1)$ are equal.

5. $Parti(\alpha)$ where α represents a person or a trusted server and intruder is not a participant: $\neg Parti(i)$.

3.2.5 Irregular Case

In following tasks, it is assumed that the case of falsification in each kind of data is unique because participants check only whether received data is same as that of participants'.

1. Add $Send(p_1, p_2, x_1), \ldots, Send(p_1, p_2, x_n)$ in $NP^1_j, \ldots, NP^n_j$, respectively if a participant sends data selectively to another.

2. If $Send(i, p, x)$ is deduced, analysts add $Recv(p, x)$ in NP^i_j.

3. If $Send(p_1, p_2, x)$ is deduced, analysts add $Recv(p_2, x)$ in NP^1_j and the logical formulas generated in task 5.

4. If $Send(p, i, x)$ is deduced, analysts add logical formulas that are generated in task 5.

5. In order to generate logical formulas that represents an intruder sends data with falsification, analysts do three tasks.

   a. Enumerate participants that receive any data after step j If there is no participant, tasks of forward reasoning is over.

   b. Generate logical formulas which data is sent in the step by replacing individual variable p_i into individual constant and then replace each data of falsification (The data that participants don't have can be falsified). If any data cannot be replaced, continue to next participant.

   c. Generate logical formulas $Send(i, p', data_1)$, $\ldots$, $Send(i, p', data_n)$, and add those formulas with $Recv(i, x)$ in $IP^1_j, \ldots, IP^n_j$, respectively.

Figure 3.1 shows an overview of branches by cases that participants and an intruder sends data. In the above conditions of falsification, the complexity of the tasks is increased by the number of logical formulas IP^m_n. If there is a case that a participant sends data selectively to another, the number of logical formulas NP^m_n also increase the complexity. However, the task of forward reasoning can be finished by the finite time because the case of falsification in each kind of data is unique.

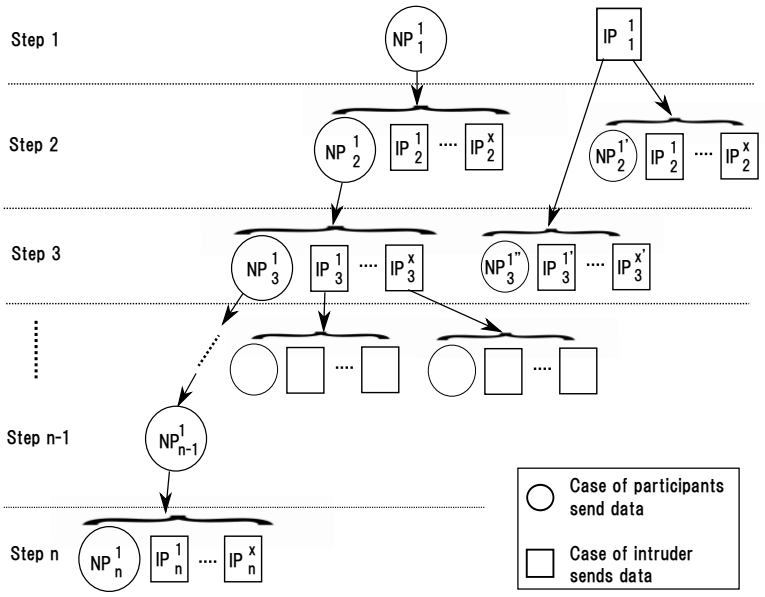

Figure 3.1: Branches by cases that participants and an intruder sends data.

3.3 Forward Reasoning

Analysts use FreeEnCal [20] to perform forward reasoning automatically. Analysts use generated logical formulas such as NP_i^j, $IP_i^{j'}$, CO and deduced logical formulas until this step (DF) as input of FreeEnCal. As the result, new logical formulas (DF_{new}) deduced, and DF_{new} is added in DF.

Therefore, if $IP_i^{j'}$ does not include data that participants get (represented as $Get(p, x)$), and should be received in step i, $IP_i^{j'}$ cannot be used as input of FreeEnCal. If DF is the result of forward reasoning about the last step of the target protocol, NP_i^j cannot be used as input of FreeEnCal. If neither NP_i^j or $IP_i^{j'}$ cannot be inputted, forward reasoning in the case is completed. If all cases of forward reasoning are completed, analysts perform tasks of analysis for each DF.

3.4 Analysis

Analysts check whether formulas that represent successful attacks are included in DF.

If formulas that represent successful completion of the protocol, and an intruder gets secret data, or any participant received falsified data are included in deduced logical formulas, it can be said that attacks by an intruder succeed in the target cryptographic protocol.

3.5 Case Studies

To confirm the effectiveness of the extended method, we select secret splitting protocols [3] as cases because secret splitting protocols have feature $F1$, $F2$, $F4$, and $F5$, and coin

flips protocols which has $F3$ [13] as Table 3.1 shown in chapter 3.

Secret splitting protocols are steps to split secret information into several parts of data, and prevent restoring secret information without collecting each split data. The protocol can prevent leakage of secret information because each split data do not have meaning, and secret data cannot be restored by only split data. In the protocol, a trusted server previously has secret information.

At first, we formalized the specification of each protocol. After that, we performed forward reasoning as the premises. Finally, we detect fatal action of attacks from deduced logical formulas by forward reasoning. If formal analysis based on the extended method can deduce logical formulas that represent successful attacks in protocols it can be said that formal analysis method with reasoning is effective for various cryptographic protocols.

we prepared two secret splitting protocols as the case of formal analysis. In these specification of secret splitting protocols, $data1$ and $data2$ are split data to send participants of protocol. Specification of first protocol is below.

Specification of the first protocol

Step 1 $A \rightarrow S : A, B$

Step 2 $S \rightarrow A : \{data1, B\}_{K_{AS}}$

Step 3 $S \rightarrow B : \{data2, A\}_{K_{BS}}$

The first protocol has a flaw. When a trusted server S checks two identifiers, there are two conditions, S checks individually whether each identifier is correct, or checks whether two identifiers are equal. If a trusted server S only checks individually whether those sent identifiers are correct, without checking whether sent two identifiers in step 1 are equal or not, it is possible that an intruder can receive split data $data1$ and $data2$. As the result, the intruder can get secret data from got split data $data1$ and $data2$ by falsifying identifiers that are sent in step 1. Concretely, the intruder can perform this attack by following steps. In following steps, $I(X)$ means that "an intruder receives data with pretending to be X."

An attack on first protocol

1. $A \rightarrow I(S) : A, B$
2. $I(A) \rightarrow S : I, I$
3. $S \rightarrow I : \{data1\}_{K_{IS}}$
4. $S \rightarrow I : \{data2\}_{K_{IS}}$

We prepared second protocol that is different from first protocol, S starts to send data to each participant. Specification of second protocol is below.

Specification of second protocol

1. $S \rightarrow A : \{data1, B\}_{K_{AS}}$
2. $S \rightarrow B : \{data2, A\}_{K_{BS}}$

The second protocol has a flaw. In behavior of an intruder based on Dolev-Yao model [29], if an intruder knows old data that is sent and received in the previous communication process, the intruder can falsify sent data into old data that the intruder knows without participants recognizing. Concretely, the intruder can perform this attack by following steps. In following steps, $data1'$ and $data2'$ are old split data to be sent and received in the previous communication process.

Attack on second protocol

1. $S \rightarrow I(A) : \{data1, B\}_{K_{AS}}$
2. $I(S) \rightarrow A : \{data1', B\}_{K_{AS}}$
3. $S \rightarrow I(B) : \{data2, A\}_{K_{BS}}$
4. $I(S) \rightarrow B : \{data2', A\}_{K_{BS}}$

It is a case that both $data1'$ and $data2'$ are falsified. On the other hand, there is a case that only $data1'$ or $data2'$ is falsified.

Attack on second protocol (in case that $data1$ is falsified)

1. $S \rightarrow I(A) : \{data1, B\}_{K_{AS}}$
2. $I(S) \rightarrow A : \{data1', B\}_{K_{AS}}$
3. $S \rightarrow B : \{data2, A\}_{K_{BS}}$

Attack on second protocol (in case that $data2$ is falsified)

1. $S \rightarrow A : \{data1, B\}_{K_{AS}}$
2. $S \rightarrow I(B) : \{data2, A\}_{K_{BS}}$
3. $I(S) \rightarrow B : \{data2', A\}_{K_{BS}}$

As the result of detecting the secret splitting protocols,

- Get(i, data1)
- Get(i, data2)

have been deduced that means success of an attack because an intruder could get each split data $data1, data2$ so that the intruder can get secret data.

- Recv(s, data(id(i), id(i)))

- Send(s, i, enc(symk(i,s), data1))
- Send(s, i, enc(symk(i,s), data2))

These three formulas equal that the intruder can get secret data.

4 logical formulas are included in deduced logical formulas which mean that participants get the falsified data because the received split data has no meaning.

- Recv(a, enc(symk(a,s), old(data1), id(b)))
- Recv(b, enc(symk(b,s), old(data2), id(a)))
- Get(a, data(old(data1), id(b)))
- Get(b, data(old(data2), id(a)))

Coin flips protocols are steps to fairly perform coin-flip for participants. At first, a participant A sends data with encrypting that receiver B cannot decrypt. After that, B sends next data the represents the received value of data is odd or even (corresponding to front or back of the coin). Finally, A sends the firstly sent data to B in order to check whether the sent data is correct. In coin flips protocols, A must not previously know the result that received data is odd or even, and B must not previously know whether B 's sent data is odd or even.

We prepared one coin flips protocol as the case of formal analysis. In following specification of coin flips protocol, $f(x)$ is one way function of sent data x, and odd, $even$ are data that is corresponding to front or back of the coin. Specification of the coin flips protocol is below.

Specification of the coin flips protocol

Step 1 $A \rightarrow B : \{f(x), A\}_{K_{AB}}$

Step 2 $B \rightarrow A : \{odd\}_{K_{AB}} / \{even\}_{K_{AB}}$

Step 3 $A \rightarrow B : \{x\}_{K_{AB}}$

We performed formal analysis of the coin flips protocol based on the extended method. As the result, formula that represent flaw was detected in deduced logical formulas. However, the protocol is generally believed secure because the probability of the successful attack is low. Therefore, the extended method cannot judge the security of the protocols with $F3$.

Therefore, we can apply the extended method to secret splitting protocols or other protocols which satisfy the feature F1, F2, F4, and F5 because we find out flaws by using the extended method. In the future, we will analyze other cryptographic protocols to confirm the effectiveness of the extended method.

Chapter 4

Improvement of Flaw Analysis Criteria

To clarify how to analyze the deduced formulas, we organized fine-grained security properties as flaw analysis criteria according to the definition of flaw. We used the flaw analysis criteria to analyze the Needham Schroeder protocol and demonstrated that the proposed flaw analysis criteria are capable to find out flaws of cryptographic protocols.

4.1 Flaw Analysis Criteria

4.1.1 Definiton of Flaws

In a certain application environment, the designed cryptographic protocol should meet security requirements. A cryptographic protocol has flaws if the protocol designer's definition and description of the security requirements are incomplete, or the designed cryptographic protocol does not satisfy all the security requirements. Flaws can exist in the cryptographic algorithm, the cryptographic techniques for implementing the algorithm and protocol, and a cryptographic protocol itself. In this paper, we only consider the flaws in the protocol itself.

In the premise that the security requirements are complete and correct, if the protocol does not meet the security requirements, we consider that the cryptographic protocol to be flawed. Security requirements can be expressed as the extraction or combination of many basic security properties. In other words, a cryptographic protocol that does not satisfy security properties corresponding to security requirements leads to the generation of flaws. Therefore, security properties are treated as criteria for determining whether flaws exist.

4.1.2 Security Properties of Cryptographic Protocols

Confidentiality

Confidentiality of cryptographic protocols means that secret information is not obtained by unauthorized entities [42]. It classified into general confidentiality, strong confidentiality, forward confidentiality, and know-key confidentiality.

1. General confidentiality

Secret information of a cryptographic protocol transmitted cannot be leaked to unauthorized entities [1][2].

2. Strong confidentiality

   Unauthorized entities cannot find the difference when the value of the $SecretItem$ changes in a cryptographic protocol [14]. In other words, if the intruder cannot guess or discover the changed value, regardless of whether the $SecretItem$ is instantiated into any data, it can be considered that the intruder does not obtain any information of the $SecretItem$ and the cryptographic protocol satisfies the strong confidentiality.

3. Forward confidentiality

   Even if the long-term key leaks to the intruder, he/she cannot obtain the session key of a certain communication [42].

4. Know-key confidentiality

   Even if the session key is compromised, the intruder cannot find other keys through the session key [4].

General confidentiality and strong confidentiality mean that secret information is not disclosed during the operation of the protocol. And the confidentiality of strong confidentiality is stronger than the general confidentiality.

Forward confidentiality and know-key confidentiality focus on key-related protocols, considering the impact of secret key leakage on secret information. Forward confidentiality mainly provides perfect confidentiality over past encrypted data and know-key confidentiality restricts the security hazard generated by the session key to the current communication.

Authentication

1. Entity authentication

   An entity authentication assures one participant (through an acquisition of corroborative evidence) of both the identity of the other participant involved and that the other participant was active at the time the evidence was created or acquired [42]. In other words, entity authentication should authenticate the identity of the communicating participants.

2. Data authentication

   Data authentication provides for one participant which receives a message assurance (through corroborative evidence) of the identity of the participant which originated the message [42]. It is the process of confirming the origin and integrity of data. First, authenticate whether the getting data are from the correct entity. Second, validate whether the data has been altered in an unauthorized manner since the time it was created, transmitted, or stored by an authorized source.

The security properties of authentication ensure the authenticity of the communication. The main difference between the two properties is that data authentication provides no timeliness guarantee (the authenticated data may be old), while entity authentication implies actual communication with an associated verifier during execution of the current run of the protocol [51].

Fairness

Fairness means that after the completion of the cryptographic protocols, any participants of the protocols has sufficient evidence to resolve possible disputes in the future. And at any stage of the operation of the cryptographic protocol, any participant will not have more privilege in the operation. It includes implementation fairness, acquired fairness and retrospective fairness [5].

1. Implementation fairness

   Any participant has the same control over the execution of the cryptographic protocol. In other words, the participants have the right to choose whether to continue or give up the execution of the protocol step and to exercise such rights without affecting the rights of other participants.

2. Acquired fairness

   In the case of normal termination of the cryptographic protocols, participants can be guaranteed the information they should have obtained. In the case of abnormal suspension, all participants did not receive anything.

3. Retrospective fairness

   No participant can escape the responsibilities associated with the exchange of information.

The above properties describe the fairness to be satisfied during the entire process of the cryptographic protocols. And they are independent throughout the cryptographic protocol.

Freshness

The fresh identifier was not used before it was generated, and all the participants believed that the identifier is also fresh.

Non-Repudiation

The participants of the cryptographic protocols must be responsible for their own actions, the sender cannot deny the messages they have sent, and the receiver cannot deny any messages received [38].

Anonymity

Participants can observe the occurrence of a series of events but cannot know who made them. Anonymity is divided into the following two forms [48].

1. Sender Anonymity

   Anonymity first provides the sender's anonymous protection, and the sender's identity information and address should be hidden.

2. Receiver Anonymity

   Similarly, anonymity should provide the receiver's anonymous protection, and the receiver's identity information and address should be hidden.

Atomicity

The concept of atomicity stems from the theory of databases, which means that a logical unit of a database consists of a series of operations. This group of operations is either executed, the state changes in a consistent manner, or the operation is not performed, and the state does not change. Tygar [54] proposed the concept of atomicity of e-commerce protocols, dividing the atomicity into three levels and being upwardly compatible, that is, the latter contains the former.

1. Money Atomicity

   Funds are conserved before and after e-commerce protocols occur, and funds are neither created nor disappeared in e-commerce. The reduction in customer money is equal to the increase in business money.

2. Goods Atomicity

   Satisfying the goods atomicity must satisfy the money atomicity. Guarantee the customer receives the goods only if the merchant receives the payment.

3. Certified Atomicity Satisfying the certified atomicity must satisfy the goods atomicity and money atomicity. It is necessary to confirm the goods purchased by the customer and the goods sent by the merchant.

In summary, the above is the security properties that cryptographic protocols should meet. It is true that a good cryptographic protocol should also satisfy the properties as high efficiency, privacy, and robustness, but how to accurately define and describe these properties is very difficult now. So we do not deeply analyze these properties for the time being.

4.2 Case Studies

In the previous section, we enumerated the security properties of cryptographic protocols. In this section, we explained the correctness and necessity of those security properties by analyzing some specific cryptographic protocols. We analyzed Needham Schroeder protocol [44] and anonymous atomic transaction protocol [16].

First, define some symbols used in the explanation of Needham Schroeder protocol and anonymous atomic transaction protocol.

- A, B, S

  Identifiers of participant A, B, S

- C, B, M, L

  Identifiers of participant custom, bank, merchant, transaction log

- I

  Identifier of an intruder.

- $K_{X_1X_2}$

  The symmetric key between participants X_1 and X_2.

- SK

  Session key of a cryptographic protocol.

- PK_X

  The public key of a participant X.

- K_X

  The long-term key (private key) of a participant X.

- Sig_X

  The signature of a participant X.

- N_X

  Nonce of a participant X.

- $\{Y_1, Y_2, \cdots, Y_z\}_K$

  Encrypted or decrypted $Y_1, Y_2, \cdots, Y_z$ by key K. If the key is a symmetric key, the symbol means encrypted or decrypted data. If the key is an asymmetric key, the symbol only means encrypted data.

- $\{Y\}_{Sigx}$

  Data Y with the signature of participant X.

4.2.1 Case Study in Needham Schroeder Protocol

Needham Schroeder protocol is an authentication and key exchange protocol [44]. The function of this protocol is to complete the mutual authentication between participants A and B and obtain the session key distributed by S.

Specification

1. $A \rightarrow S : A, B, N_a$
2. $S \rightarrow A : \{N_a, B, SK, \{SK, A\}_{K_b}\}_{K_a}$
3. $A \rightarrow B : \{SK, A\}_{K_b}$
4. $B \rightarrow A : \{N_b\}_{SK}$
5. $A \rightarrow B : \{N_b - 1\}_{SK}$

This protocol has a flaw [27]. If the intruder intercepts the data sent by participant A to B in the third step and disguises as B and sends a random number N_i in the same form as $\{N_b\}_{SK}$in the fourth step, then A sent the $\{\{N_i\}_{SK} - 1\}_{SK}$ to the intruder. A thought that B had obtained and confirmed the session key, but in fact, B did not participate in the execution of the protocol.

The first attack

1. $A \rightarrow S : A, B, N_a$
2. $S \rightarrow A : \{N_a, B, SK, \{SK, A\}_{K_b}\}_{K_a}$
3. $A \rightarrow I(B) : \{SK, A\}_{K_b}$
4. $I(B) \rightarrow A : N_i$
5. $A \rightarrow I(B) : \{\{N_i\}_{SK} - 1\}_{SK}$

According to the definition of security properties, we can find that this protocol does not satisfy strong confidentiality, entity authentication and data authentication. In the fourth step and fifth step, the intruder receives N_i and $\{\{N_i\}_{SK} - 1\}_{SK}$. Based on these two data, the intruder can guess the value of SK and verify it [39], so that the protocol does not satisfy the strong confidentiality. In the fourth step, participant A does not know if the entity communicating with him/her is real participant B because A have no idea to check the identity of the entity, so it does not satisfy the entity authentication. Participant A also cannot authenticate whether the received message actually came from B. Since the message was falsified in the fourth step, the protocol does not satisfy the properties of data authentication.

There is also another flaw in this protocol [27]. If the intruder gets access to K_a, he/she can record S's messages to A. Then, the intruder can pretend to be A to communicate with B. Also, participant B cannot judge whether the received message in the fifth step is fresh.

The second attack

1. $A \rightarrow S : A, B, N_a$
2. $S \rightarrow I(A) : \{N_a, B, SK, \{SK, A\}_{K_b}\}_{K_a}$
3. $I(S) \rightarrow A : \{N_a, B, SK, \{SK, A\}_{K_b}\}_{K_a}$
4. $A \rightarrow I(B) : \{SK, A\}_{K_b}$
5. $I(A) \rightarrow B : \{SK, A\}_{K_b}$
6. $B \rightarrow I(A) : \{N_b\}_{SK}$
7. $I(A) \rightarrow B : \{N_b - 1\}_{SK}$

According to the definitions of security properties, we can find that the reason for this flaw is that the protocol does not satisfy freshness and forward confidentiality. Leakage of participant A's long-term key K_a leads to the leakage of SK, that is, the protocol does not meet the forward confidentiality. For participant B, the message "$\{SK, A\}_{K_b}$" can only indicate that the operation of this step is fresh, and the fresh identifier SK is not authenticated by the generating participant S, so participant B cannot judge whether the message is a replay message or falsified message. Therefore, the protocol also does not meet freshness.

In summary, the security properties that Needham Schroeder protocol should be met is confidentiality, authentication, and freshness. Even if one of these properties is not met, it can create flaws that can be exploited by intruders.

4.2.2 Case Study in Anonymous Atomic Transaction Protocol

In the case study, we do not analyze confidentiality, authentication, and freshness, and focus on fairness, non-repudiation, anonymity and atomicity. Anonymous atomic transaction protocol is an e-commerce protocol that for two or more participants to implement online transaction functions [16]. This protocol introduces the concept of transaction log, independent of the customer, the merchant, and the bank, the fourth party. Specifically divided into withdrawal protocol and purchase protocol.

The customer has an account in the bank. He/she can withdraw the anonymous token from the bank. Only the bank can issue the token and the bank cannot track the identity of the customer through the token. However, if the customer re-uses the token, the bank will check it out and reveal the identity of the customer.

Withdrawal protocol

The protocol for customer C to withdraw anonymous tokens from bank B is as follows.

Specification

1. $C \rightarrow B : \{\{C, Q, V\}_{Sigc}\}_{K_b}$
2. $B \rightarrow C : \{\{Token(Q, V)\}_{Sigb}\}_{K_c}$

The customer randomly generates a pair of withdrawal keys Q and q. C is the identifier of customer, V indicates withdrawal amount. Customer signed the information, then encrypted with the public key of the bank and sent to the bank. The bank checks the customer's account and generates $Token(Q, V)$ if there is a balance. The bank also generates a database to manage the $Token(Q, V)$. The database is as follows.

Q	V	Feature	Pointer to customer
...	...	...	...
...	...	...	...

Table 4.1: The database of Token

When a merchant comes to the bank to exchange tokens, the bank first checks the corresponding feature, and if it has not been consumed, the value V of the token is sent to the merchant's account and the feature is marked as spent. If the feature shows that it has been spent, it means that the cost is repetitive. The bank will find the customer's identity based on the pointer.

After the bank generates the $Token(Q, V)$, the signed $Token$ is encrypted by the customer's public key and sent to the customer.

The customer receives the message and gets the anonymous token. If there is no recurring spending, banks and merchants cannot know the customer's identity. Since only the feature value in the database is negative, the pointer to the client's identity is accessed. Thus, the protocol satisfies the sender anonymity.

Purchase protocol

This protocol describes customer C using anonymous $Token(Q, V)$ to purchase.

Specification

1. $C \rightarrow M : \{Goods\}_{K_m}$
2. $M \rightarrow C : \{\{N_m, Contract, \{Goods\}_{SK}\}_{Sig_c}\}_{K_c}$
3. $C \rightarrow B : \{\{Token(Q, V), \{N_m, Deadline, M, L\}_{Kq}\}_{K_b}$
4. $B \rightarrow M : \{N_m, Deadline, M, L, V\}_{Sig_b}$
5. $M \rightarrow L : \{N_m, Deadline, SK\}_{Sig_m}$
6. $L \rightarrow C : \{SK\}_{Sig_l} / \{Fail\}_{Sig_l}$
7. $L \rightarrow B : \{SK\}_{Sig_l} / \{Fail\}_{Sig_l}$
8. $L \rightarrow M : \{SK\}_{Sig_l} / \{Fail\}_{Sig_l}$

Detail of each step of the protocol is as follows. In step 1, customer C sends the selected $Goods$ encrypt by the merchant's public key to the merchant M.

Then the merchant generates a message for the goods, encrypts it with the customer's public key and sends it to the customer. N_m represents the transaction number, $Contract$ includes the goods description and price, and SK represents the session key of contract.

In step 3, after the customer confirms the message from the merchant, he/she sends the bank authentication information and authorizes payment in $Token(Q, V)$. $Deadline$ is the validity period of $Token(Q, V)$. L means the transaction log. After the deadline, the payment will be canceled and the customer can use the token for other consumption or refund to the bank.

In step 4, the bank decrypts the authentication message from the customer, verifies the validity of $Token(Q, V)$ and check whether there is repeated consumption or whether it exceeds the deadline. After the inspection is completed, the authentication message $\{N_m, Deadline, M, L, V\}_{Sig_b}$is sent to the merchant.

In step 5, merchant confirmation message is issued by the bank, verification transaction number N_m, commodity price V, expiration date $Deadline$ and transaction log L. Then send the signed information $\{N_m, Deadline, SK\}_{Sig_m}$ to the transaction log.

In the next steps, transaction log L confirms commitment message from merchants, records transaction number N_m, expiration date $Deadline$, transaction session key SK, and timestamps at the time of receipt. When it is determined that everything is valid, the transaction log L sends the signed transaction session key SK to customer C, bank B, and merchant M. If the $Deadline$ expires, then the transaction log L sends the failed transaction $Fail$ to customer C, bank B, and merchant M.

The customer C receives the message from the transaction log L and uses the transaction session key SK to decrypt the encrypted goods $\{Goods\}_{SK}$ received in the second step to obtain the goods. Then compare the goods and $Contract$ to confirm consistency. If inconsistent, the customer can claim against the merchant with the$Contract$, $\{Goods\}_{SK}$ and $\{SK\}_{Sig_l}$. If the customer receives $Fail$ message, $Token(Q, V)$ can be reused.

The bank B receives $\{SK\}_{Sig_l}$ from the transaction log L indicating that the transaction was successful and the bank transferred the value of $Token(Q, V)$ to the merchant ' s account. If $Fail$ message is received, the transaction has failed and the bank marks the $Token(Q, V)$ in the database as not consumed.

The merchant M receives $\{SK\}_{Sig_l}$ from the transaction log L indicating that the customer C had obtained the goods, and the merchant 's account should have added value V. If the account does not increase, merchants M can use $\{SK\}_{Sig_l}$ and the message received in step 4 to arbitrate. If the merchant M receives $Fail$ message, it means that the transaction failed, the customer cannot get the goods, and the merchant 's account does not increase in value.

In summary, after the protocol is implemented normally, all participants involved can obtain the necessary information. If there is an exception, all participants get nothing, which meets the properties of fairness.

In the withdrawal protocol, the bank checks whether tokens are consumed. If there is no recurring spending, bank and merchant cannot know the customer's identity. Since only the feature value in the database is negative, the pointer to the client's identity is

Properties	Needham Schroeder	Anonymous atomic transaction
Confidentiality	○	
Authentication	○	
Fairness		○
Freshness	○	
Non-Repudiation		○
Anonymity		○
Atomicity		○

Table 4.2: The Properties That Protocols Meet

accessed. thus, the protocol satisfies the sender anonymity.

From the description of the anonymous atomic transaction protocols, each commit of customers, merchants and banks are protected by digital signatures, so that they cannot deny or reject the information sent or received, which satisfies the security properties of non-repudiation.

It can be seen that the customer 's reduced money is exactly equal to the merchant 's increased money, so it satisfies the properties of money atomicity. If the customer pays the money, he will be able to get the goods and if he gets the goods, he must pay the money so that it satisfies the goods atomicity. Only the customer can confirm whether the received goods are the same as the order, but the merchant cannot confirm it, so it does not meet the certified atomicity.

4.2.3 Summary of Case Studies

By analyzing the Needham Schroeder protocol and anonymous atomic transaction protocol, it is clear that the enumerated security properties are indeed the cryptographic protocol should satisfy. When some properties are not satisfied by a cryptographic protocol, the protocol to be flowed. The table below shows the properties of each two protocols. It indicates that the seven security properties are necessary for cryptographic protocols and can be the criteria for determining whether flaws exist.

Chapter 5

Improvement of Detecting Flaws

To detect more types of cryptographic protocol flaws, we extended the intruder's behavior to describe guessing attacks and external replay attacks and proposed a new method to perform forward reasoning to test the flaws related to non-repudiation and fairness caused by participants' deception. We also took the Needham Schroeder protocol and ISI protocol as cases to perform security analysis. By the result of analyzing, it is proved that the improved formal analysis method with reasoning is effective to detect the flaws of guessing attacks, external replay attacks and the flaws related to non-repudiation and fairness.

5.1 Limitations of Detecting Flaws

Although the formal analysis method with reasoning can be used to detect flaws in a variety of cryptographic protocols, there are still some limitations, that is, the types of flaws that can be detected are limited. The reason is that the Dolev-Yao model can only describe the behavior of an outside intruder but does not consider the mutual deception of the participant inside the protocol. Therefore, all flaws related to internal attacks cannot be found out.

5.2 Extending Formal Analysis Method with Reasoning by Adding Participant's Behaviors

5.2.1 Basic Notions and Security Properties

Participant is an authorized entity who executes the cryptographic protocols. *Participant* is divided into two types. One is an honest participant who sends or receives data in strict accordance with the steps of a cryptographic protocol. The other is a dishonest participant who may lie in the execution of a cryptographic protocol or not execute the cryptographic protocol at all, trying to impersonate or deceive the other participant to achieve various illegal purposes. In this paper, we assumed that participant A and B may be dishonest participants, while CS (currency server) and TTP (trusted third party) are always honest participants.

The execution of each message of a cryptographic protocol is related to the security of the environment. If the message is not executed or is not executed correctly, we consider that the message is interruptible. If a message should be sent by a dishonest participant, it is interruptible because the dishonest participant may choose to stop sending a message for his own benefit. If a message should be sent by an honest participant, the message is uninterruptible in a situation of secure environment because an honest participant will always follow the protocol process. If the environment is not secure, the message sent by the honest participant may be lost, so that the message is interruptible. This thesis only considers the interruption in a secure environment.

Non-repudiation [35][36][38] and fairness [5][32][38] are the security properties of cryptographic protocols. Non-repudiation means that the participants of a cryptographic protocol should be responsible for their actions, the sender cannot deny the messages they have sent, and the receiver cannot deny any messages received. In the event of a dispute, a participant can provide the necessary evidence to protect its own interests. Non-repudiation is achieved by the sender having the receiver's non-repudiation evidence and the receiver having the sender's non-repudiation evidence [52][58]. The sender's non-repudiation evidence is used to prove that the sender did send the message, and the receiver's non-repudiation evidence to prove that the receiver did receive the message.

Fairness means that at any stage of the operation of a cryptographic protocol, any participant will not have more privilege in the operation. It includes implementation fairness, acquired fairness and retrospective fairness [5][47]. Implementation fairness means any participant has the same control over the execution of the cryptographic protocol. In other words, the participants have the right to choose whether to continue or give up the execution of the protocol step and to exercise such rights without affecting the rights of other participants. Acquired fairness means in the case of normal termination of the cryptographic protocols, participants can be guaranteed the information they should have obtained. In the case of abnormal suspension, all participants did not receive anything. Retrospective fairness means no participant can escape the responsibilities associated with the exchange of information. In this thesis, we focused on the realization of acquired fairness, which is also the most important aspect of fairness.

5.2.2 Formalization

Overview of Formalization

In the previous formal analysis method with reasoning, analysts formalize the "behaviors of participants", "behaviors of an intruder", "common behaviors among participants and an intruder" and "irregular case" as targets of formalization [57]. In the extended method, to detect the flaws that caused by dishonest participants, we have added "confirm behaviors" and some new tasks in "common behaviors". Below we only provide a formal description of behaviors to perform non-repudiation and fairness flaws detection.

Analysts formalize "behaviors of participants", "common behaviors" and "confirm behaviors" as targets of formalization. "behavior of participants" means a set of rules of participants." "common behaviors" represents rules except for the behaviors of sending and receiving data by participants. "confirm behaviors" represents a set of rules that

participants judge whether the signatures or keys are true.

To formalize the behaviors above, we used the definition of predicates, functions and individual constants in section 3.2.1 and added following predicates and individual constants that represents participants' behavior or data in cryptographic protocols.

Predicates

- $Cof(p_1, p_2, x)$: p_1 confirms that p_2 has responsibility of x.

Constants

- ttp: Trusted third sever.

In addition, there are uniquely defined functions and individual constants that are assigned to each data in a cryptographic protocol.

Behaviors of Participants

Behaviors of participants are to describe the specification of a cryptographic protocol. Generally, in specification of cryptographic protocols, sending and receiving in step M are represented as $M.X_1 \rightarrow X_2 : Y_1, Y_2, \cdots, Y_z$, it means that X_1 sends data $Y_1, Y_2, \cdots, Y_z$ ($z \in \mathbb{N}$) to X_2 in step M.

1. Represent participants' behaviors in each step of a cryptographic protocol by formulas. p_i and x_i are individual variables, n and m are the number of sent or received data.

   a. As step 1 of the protocol,

   $$Start(p_1, p_2) \Rightarrow Send(p_1, p_2, data(x_1, \ldots, x_n)) \tag{5.1}$$

   means "if a cryptographic protocol starts with p_1 and p_2, p_1 sends data $x_1, \ldots, x_n$ to p_2".

   b. As step 2 of the protocol,

   $$Recv(p_1, data(x_1, \ldots, x_m)) \Rightarrow (Parti(p_1) \Rightarrow Send(p_1, p_2, data(x_1, \ldots, x_n))) \tag{5.2}$$

   means "if a participant p_1 receives data, p_1 sends the next data to p_2." p_i, x_i are individual variables, and n, m are the number of sent or received data.

2. Replace individual variables p_1 and p_2 in formulas (5.1) and (5.2) with ttp or p_i respectively in the previous task. For example, if sender of corresponding step is a third trusted party ttp, individual variable p_1 is replaced with the individual constant ttp.

3. Replace individual variables $x_1, \cdots, x_n$ in formulas (5.1) and (5.2) with terms according to following rules corresponding step of the specification.

a. If sent data Y_i is not encrypted, substitute a function $f(p_i)$ or$f(ttp)$ respectively or an individual variable that is uniquely defined.

b. If Y_i is incremented data, substitute $plus(x_i')$. x_i' is replaced as well as previous task 3-a.

c. If Y_i is encrypted data, substitute $enc(k, x_1', \cdots, x_n')$ and replace k depending on key types such as public key or symmetric key.

d. If Y_i is signed data, substitute $sig(p_i, x_1', \cdots, x_n')$ and replace p_i with $p_1, p_2, \ldots, p_n$ or ttp.

4. In part of formulas $A_1 \Rightarrow A_2$ (A_1, A_2 is formulas), if a variable is included only in A_1 or A_2, define an individual constant and replace the variable into the constant.

5. Add quantifier $\forall$ corresponded to individual variables k, x_i, and p_i in those formulas.

6. Generate a formula $Start(p_1, p_2)$ with substituting an individual constant of participants to p_1 and p_2.

Common Behaviors

Common behaviors mainly describe implicit behaviors such as encryption or decryption behaviors of participants.

1. Generate $\forall p((Recv(p, recvdata_1) \wedge \cdots \wedge Recv(p, recvdata_n) \Rightarrow Recv(p, recvdata')))$ represents if p receives multiple data, p receives another data.

2. Generate the formula that means if p receives data encrypted by p's symmetric key or public key, p gets original data. $\forall p_1 \forall p_2 \forall x_1 \ldots \forall x_n(Recv(p_1, enc(symk(p_1, p_2), x_1, \cdots, x_n)) \Rightarrow Recv(p_1, data(x_1, \cdots, x_n)))$, and $\forall p \forall x_1 \ldots \forall x_n(Recv(p, enc(pk(p), x_1, \cdots, x_n)) \Rightarrow Recv(p, data(x_1, \cdots, x_n)))$

3. Generate the formula that means if p receives data encrypted by a session key that p knows, p gets original data. $\forall p_1 \forall x_1 \ldots \forall x_n(Recv(p_1, enc(sesk(p_1), x_1, \cdots, x_n)) \Rightarrow Recv(p_1, data(x_1, \cdots, x_n)))$

4. $\forall p_1 \forall p_2(Eq(symk(p_1, p_2), symk(p_2, p_1)))$ represents $symk(p_1, p_2)$ and $symk(p_2, p_1)$ are equal.

5. $Parti(\alpha)$ where α represents a person or the third trusted server.

Confirm Behaviors

Confirm behaviors are the basis for the participants to judge whether the received message is correct or not.

1. Generate formulas of the public key other participants that each participant have confirmed before the cryptographic protocol executing. $\forall p_1 \forall p_2(Cof(p_1, p_2, pk(p_2)))$, which means p_1 have confirmed that p_2 has responsible for the $pk(p_2)$.

2. If participnat p_1 receives the data $\{x\}_{sig_{p_2}}$, and p_1 confirms that p_2 has the public key which can decrypt the signiture, then p_1 can confirm p_2 has responsibility of data x. $\forall p_1 \forall p_2 \forall x(Recv(p_1, \{x\}_{sig_{p_2}}) \wedge Cof(p_1, p_2, pk(p_2)) \Rightarrow Cof(p_1, p_2, x))$

3. If participnat p_1 receives the data $\{x\}_k$, and p_1 confirms that k is the key of p_2, p_1 can confirm p_2 has responsibility of data x. $\forall p_1 \forall p_2 \forall x(Recv(p_1, \{x\}_k) \wedge Cof(p_1, p_2, k) \Rightarrow Cof(p_1, p_2, x))$

4. If participnat p_1 confirms the data $\{x\}_k$, and p_1 confirms that k is the key of p_2, p_1 can confirm p_2 has responsibility of data x. $\forall p_1 \forall p_2 \forall x(Cof(p_1, \{x\}_k) \wedge Cof(p_1, p_2, k) \Rightarrow Cof(p_1, p_2, x))$

5. If participnat p_1 confirms that p_2 has responsibility to data set (x, y, z), p_1 can confirm p_2 has responsibility of each data, same if vice versa. They are represented as $\forall p_1 \forall p_2 \forall x_1 \ldots \forall x_n(Cof(p_1, p_2, \{x_1 \ldots x_n\}) \Rightarrow Cof(p_1, p_2, x_1) \wedge \cdots \wedge Cof(p_1, p_2, x_n))$ and $\forall p_1 \forall p_2 \forall x_1 \ldots \forall x_n(Cof(p_1, p_2, x_1) \wedge \cdots \wedge Cof(p_1, p_2, x_n) \Rightarrow Cof(p_1, p_2, \{x_1 \ldots x_n\}))$

6. TTP and CS are always honest participants. If participants p_1 and p_2 exchange data through TTP or CS, and p_1 confirms TTP or CS has responsibility to data m, p_1 can confirm p_2 has responsibility to data m. $\forall p_1 \forall p_2 \forall x(Cof(p_1, ttp, x_1) \Rightarrow Cof(p_1, p_2, x_1))$

5.2.3 Forward Reasoning

In the method, analysts use FreeEnCal [20] to perform forward reasoning automatically. In the extended method, the number of forward reasoning executions depends on the number of steps of the cryptographic protocol to be detected. Analysts put generated logical formulas of "common behaviors", "confirm behaviors" into FreeEncal, and put the first step of the "behaviors of participants" into FreeEnCal for the first execution of forward reasoning, and then add the formula that represents the second step of the protocol to the result of first execution of forward reasoning to perform forward reasoning the second time. Add the formula that represents each step of the protocol in turn. If the added formula is represented as the final step of the cryptographic protocol, forwrd reasoning is finished.

5.2.4 Analysis

Analysts check the deduced formulas in each execution result of forward reasoning. If in the final execution result, all participant have confirmed the data they should receive, the cryptographic protocol has no flaws of not satisfying non-repudiation. Because according to the "confirm behaviors", if one participant deceives the other in the protocol, the other participant cannot confirm the data he/she has received. If in the middle execution results, one participant has confirmed the target data but the other has not, it indicates that the cryptographic protocol has the flaw of not satisfying fairness.

5.2.5 Case Studies

To validate the extension of formal analysis method with reasoning is valid of detecting flaws of not satisfying non-repudiation and fairness, we use ISI protocol [40] and CMP1 [26] protocol as cases to analyze the security. ISI protocol is proposed by Medvinsky and Neuman, the purpose of the protocol is participant A pays participant B, then B gives the receipt to A.

The specification of the protocol is as follows. In step 1, participant A sends the symmetric key to ask the public key of B, and then B answers. Then A sends the electronic money, the service identification number to be obtained and password to B. By checking the signature of the CS, B can confirm the validity of the electronic money. In the fourth step, B transmits the electronic money to CS (currency server). Then CS will pay B in step 5. After receiving the money, B sends the receipt to A. In this protocol, CS is an honest participant. Here, $coins$, id, $password$, $transaction$, $new-coins$, $amount$, tid and $date$ are data that is uniquely defined in the protocol.

Specification

1. $A \rightarrow B : K_{ab}$

2. $B \rightarrow A : \{K_b\}_{K_{ab}}$

3. $A \rightarrow B : \{\{coins\}_{Sig_{cs}}, SK_a, id, password\}_{K_b}$

4. $B \rightarrow CS : \{\{coins\}_{Sig_{cs}}, SK_b, transaction\}_{K_{cs}}$

5. $CS \rightarrow B : \{\{new-coins\}_{Sig_{cs}}\}_{SK_b}$

6. $B \rightarrow A : \{\{amount', tid, date\}_{Sig_b}\}_{SK_a}$

This protocol has been pointed out that it does not satisfy the security of non-repudiation and fairness. In the fifth step, B receives the currency signed by CS, but at this time A does not receive the receipt. If B stops the cryptographic protocol, A has no way to get the receipt. Therefore, according to the definition of fairness, the cryptographic protocol has a flaw of not satisfying fairness. In the sixth step of the protocol, A receives the signed receipt, but A cannot confirm whether this is the signature of B, so

according to the definition of non-repudiation, the protocol has the flaw of not satisfying non-repudiation.

As the result of formal analysis by the extended method, 21 paths were generated and 2435 logical formulas were deduced. In the 18th path,

- $Cof(b, a, new - coins)$

was deduced which means that participant B can confirm that A has responsibility of $new - coins$, that is, B received the currency. In the 21st path,

- $Recv(a, data(\{amount, tid, date\}_{Sig_b}))$

was deduced which means that A only cannot confirm whether the signature is from B. Therefore, this protocol has the flaw that does not satisfy non-repudiation. Since A did not get a receipt when the currency was obtained by B on the 18th path, the protocol has a flaw that does not satisfy fairness.

In addition, if using the previous formal analysis method with reasoning to analyze ISI protocol, we will find that the protocol does not satisfy the authentication. The specific attack is as follows.

Attack

1. $A \rightarrow I(B)v : K_{ab}$
2. $I(A) \rightarrow B : K_{ab}$
3. $B \rightarrow I : \{K_b\}_{K_{ab}}$
4. $I(B) \rightarrow A : \{K_i\}_{K_{ab}}$
5. $A \rightarrow I(B) : \{\{coins\}_{Sig_{cs}}, SK_a, id, password\}_{K_i}$
6. $I(A) \rightarrow B : \{\{coins\}_{Sig_{cs}}, SK_a, id, password\}_{K_i}$
7. $B \rightarrow CS : \{\{coins\}_{Sig_{cs}}, SK_b, transaction\}_{K_{cs}}$
8. $CS \rightarrow B : \{\{new - coins\}_{Sig_{cs}}\}_{SK_b}$
9. $B \rightarrow I(A) : \{\{amount, tid, date\}_{Sig_b}\}_{SK_a}$
10. $I(B) \rightarrow A : \{\{amount', tid', date'\}_{Sig_i}\}_{SK_a}$

As a result of formal analysis, formula is

- $Recv(a, data(amount', tid', date'))$

deduced, which means that A received the falsified data. Therefore, the protocol also does not satisfy the security of authentication.

CMP1 protocol is a certified e-mail protocol. It is considered that there is no flaws related to non-repudiation and fairness.

Specification

1. $A \rightarrow B : h(m), \{K\}_{K_{ttp}}, \{\{m\}_{Sig_a}\}_k$

2. $B \rightarrow A : \{\{h(m)\}_{Sig_b}\}, \{K\}_{K_{ttp}}, \{\{m\}_{Sig_a}\}_k$

3. $TTP \rightarrow B : \{\{m\}_{Sig_a}\}_{Sig_{ttp}}$

4. $TTP \rightarrow A : \{\{h(m)\}_{Sig_b}\}, (B, m)\}_{Sig_{ttp}}$

As the result, in $DF3$, $Recv(B, m)$ is deduced, which indicates that B get the m and A does send m. At the same time, $Recv(A, h(m))$ and $Recv(A, (B, m))$ are deduced, which mean that A knows B have get m. Therefore, the protocol satisfy the non-repudiation and fairness.

In a word, it can be said that the extended method can detect flaws related to non-repudiation and fairness.

Chapter 6

Supporting Environment for Formal Analysis

6.1 Difficulties in Formal Analysis of Cryptographic Protocols

Although various supporting tools have been proposed and applied, there are still some difficulties when analysts perform formal analysis of cryptographic protocols.

Firstly, formalization process and its inverse process still need to perform manually [55] [56] . when analysts use supporting tools to perform formal analysis, formalization process should be performed at first. In particular, the properties of the cryptographic protocol should be converted into formalism acceptances or formal theorems. However, the incompleteness of informal properties makes the formalization process difficult [33], which leads to the time-consuming and error-prone problems of manual formalization. Similarly, the inverse process of formalization also needs to perform manually. In formal analysis method with reasoning, it's difficult for analysts to transform the deduced logic formulas into their natural language representation.

In addition, if analysts want to use different supporting tools to analyze a cryptographic protocol, they must master each language used by those tools, for example, Scyther's input language is loosely based on a C/Java-like syntax [25] but ProVerif's input language is a variant of the applied pi calculus [11], which increases the difficulty of using them.

Secondly, in formal analysis method with reasoning, analysis of deduced formulas by forward reasoning is also a time-consuming complex task. In the analysis process, analysts need to find possible successful attacks from millions of formulas. It's difficult for analysts to deal with massive amount of deduced formulas to find possible successful attacks.

Thus, to solve the automatic formalization problem and its inverse process of various supporting tools and provide convenient for analysts to analyze the deduced logic formulas, in other words, to support the whole processes of formal analysis for cryptographic protocols automatically, a supporting environment is necessary and urgent.

6.2 Requirement Analysis and Function Definition

To develop the supporting environment for formal analysis of cryptographic protocols, we enumerated following requirements for the supporting environment.

R1: The supporting environment should support the formalization process.

- **R1.1:** The supporting environment should provide the function of transforming informal properties of the cryptographic protocol into the language of formal analysis tool of the selected tool.
- **R1.2:** The supporting environment should provide the function of transforming logical formulas into the language of formal analysis tool.

R2: The supporting environment should support the inverse process of formalization process.

- **R2.1:** The supporting environment should provide the function of transforming the language of formal analysis tool into logic formulas.
- **R2.2:** The supporting environment should provide the function of transforming logic formulas into informal properties.

R3: The supporting environment should support the process of formal analysis.

- **R3.1:** The supporting environment should support the automatic tool for forward reasoning method.
- **R3.2:** The supporting environment should support the automatic tools for model checking method.
- **R3.3:** The supporting environment should support the automatic tools for theorem proving method.

R4: The supporting environment should support analysts to analyze the result of forward reasoning.

- **R4.1:** The supporting environment should provide the function of filtering the logical formulas.
- **R4.2:** The supporting environment should provide a function that sorts out the filtered logical formulas according to the possibility of successful attacks.

R5: The supporting environment should systematically store the inputting properties of cryptographic protocols.

- **R5.1:** The supporting environment should store the properties for each user.

R5.2: The supporting environment should store the properties of each cryptographic protocols.

R5.3: The supporting environment should store the properties for each creation times.

R6: The supporting environment should make possible for analysts to manage the properties of cryptographic protocols.

R6.1: The supporting environment should make possible for analysts to search the properties of cryptographic protocols.

R6.2: The supporting environment should make possible for analysts to modify the properties of cryptographic protocols.

R6.3: The supporting environment should make possible for analysts to delete specification of cryptographic protocols.

R7: The supporting environment should systematically store the result of formal analysis.

R7.1: The supporting environment should store the result of formal analysis for each user.

R7.2: The supporting environment should store the result of formal analysis for each cryptographic protocols.

R7.3: The supporting environment should store the result of formal analysis for each time of formal analysis.

R7.4: The supporting environment should store and manage the result of formal analysis for each divided cases in tasks of forward reasoning.

R8: The supporting environment should make possible for analysts to manage the result of formal analysis.

R8.1: The supporting environment should make possible for analysts to search the result of formal analysis by each formal method.

R8.2: The supporting environment should make possible for analysts that they can delete the result of formal analysis.

It is a hard task for analysts to perform formalization, forward reasoning or verification, and analyzing the result of formal analysis. **R1**, **R2**, **R3**, **R4** are the basic requirements of the supporting environment for formal analysis method.

It is possible that analysts analyze many cryptographic protocols. As the result, it takes lots of time to find target properties or results from many properties or results. Therefore, the supporting environment should satisfy **R5**, **R7** to save the properties and results of each cryptographic protocol.

It can be assumed that analysts improve a target cryptographic protocol from the result of formal analysis. Therefore, the supporting environment should satisfy **R6** to manage the properties of cryptographic protocols.

It can be assumed that analysts manage the result of the formal analysis of target cryptographic protocol. Therefore, **R8** should be satisfied by the supporting environment.

To satisfy the requirements above, a list of functions the supporting environment is as follows.

F1: Formalization function (satisfied **R1**)

- **F1.1:** Transform informal properties into logic formulas.
- **F1.2:** Transform logic formulas into language of formal analysis tool.

F2: Inverse formalization function (satisfied **R2**)

- **F2.1:** Transform language of formal analysis tool into logic formulas.
- **F2.2:** Transform logic formulas into informal properties.

F3: Formal analysis function (satisfied **R3**)

- **F3.1:** Forward reasoning.
- **F3.2:** Proving by model checking.
- **F3.3:** Proving by theorem proving.

F4: Filter function (satisfied **R4**)

- **F4.1:** Filter logic formulas.
- **F4.2:** Sort out the filtered logic formulas.

F5: Making directory function (satisfied **R5**,**R7**)

- **F5.1:** Make directories of the properties and results of cryptographic protocols for each user.
- **F5.2:** Make directories of the properties and results of cryptographic protocols for each cryptographic protocol.

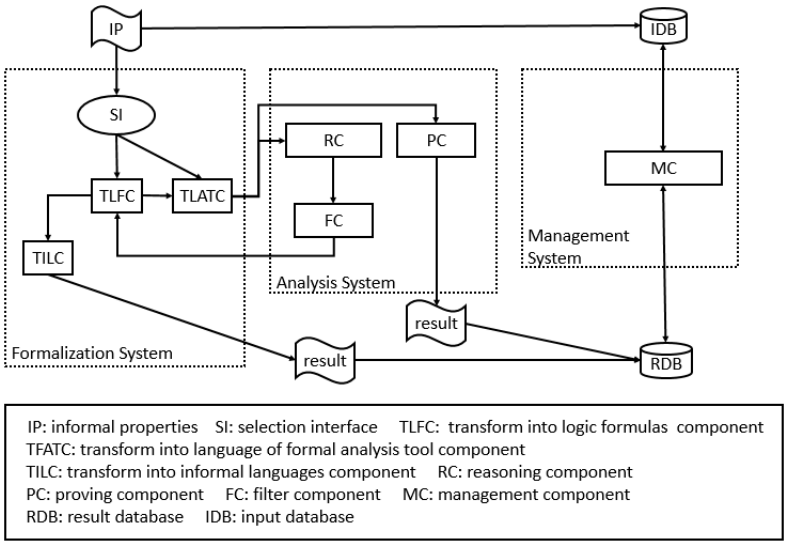

Figure 6.1: An architecture of the supporting environment

F5.3: Make directories of the properties and results of cryptographic protocols for each creation time.

F6: Properties managing function (satisfied **R6**)

F6.1: Search the properties of cryptographic protocols.

F6.2: Modify the properties of cryptographic protocols.

F6.3: Delete the properties of cryptographic protocols.

F7: Result managing function (satisfied **R8**)

F7.1: Search the result of cryptographic protocols.

F7.2: Delete the result of cryptographic protocols.

6.3 Design of the Supporting Environment

The architecture of the supporting environment is shown in figure 6.1. The supporting environment has three suv-systems that are formalization system, analysis system, and management system. It also includes two databases named result database (RDB) and input (IDB).

Formalization system mainly aims to complete the transformation between informal properties of cryptographic protocols (IP) and logic formulas, or logic formulas and the

language of formal analysis tool. In this system, there is an interface to corresponding to each formal analysis tool. The component of transforming into the logic formulas (TLFC) consists two functions, one is to transform IP into logic formulas, the other is to transform language of formal analysis tool to logic formulas. The component of transforming into the language of formal analysis tool (TLATC) also consists two functions that transform IP into language of formal analysis tool or transform logic formulas into language of formal analysis tool. Transforming into informal language component (TILC) is used to transform the filtered logic formulas into informal language to make the results clearly for analysts.

Analysis system provides the most critical step in formal analysis of a cryptographic protocol, that is, forward reasoning or proving. Therefore, in the analysis system, there are two components that are reasoning component (RC) and proving component (PC). RC corresponds to the function of performing forward reasoning and the PC corresponds to the function of performing proving of the formalized protocols. Filter component (FC) is to filter the deduced formulas of forward reasoning so that analysts can analyze them easily.

The management system aims to manage the input IP and the result of forward reasoning or proving. Analysts can delete, modify or search them by using the manage functions.

Two database RDB and IDB are used to reserve the input IP and the results of forward reasoning or proving for each user, each cryptographic protocols and each creation time.

The data processing of the supporting environment is that IP is input to the formalization system. Simultaneously, they are saved to the IDB. After selection the tools, IP will be transformed into the logic formulas or language of formal analysis tool. Then the results will work to perform forward reasoning and the results of forward reasoning will are filtered. After re-entering the filtered results to the TLFC, the filtered results will be transformed into logic formulas and at last transformed into informal language. The informal language will be saved to the RDB so that analysts can manage them conveniently by the management system. On the other hands, if the PC is started up, proving tools will work and the results of proving will also be saved to the RDB.

6.4 Usage of the Supporting Environment

Overview of the supporting environment is represented as figure 6.2.

Details process are that analysts input user name, cryptographic protocol name and properties of the cryptographic protocol as a target of formal analysis. Then select the formal analysis tool. After that, the properties will be transformed into LF (logical formulas) and LAT (language of formal analysis tool) which correspond to the formal analysis tool. If analysts would like to use the formal analysis method with proving, only formalization and proving tasks be needed. For example, when we select ProVerif, the function of formalization will put informal properties of the cryptographic protocol into ProVerif's input language, a variant of the applied pi calculus which support types [6]. Then Proverif can automatically verify whether flaws exist or not. Because the output produced by ProVerif is rather Verbtim, we do not have to do transformation work.

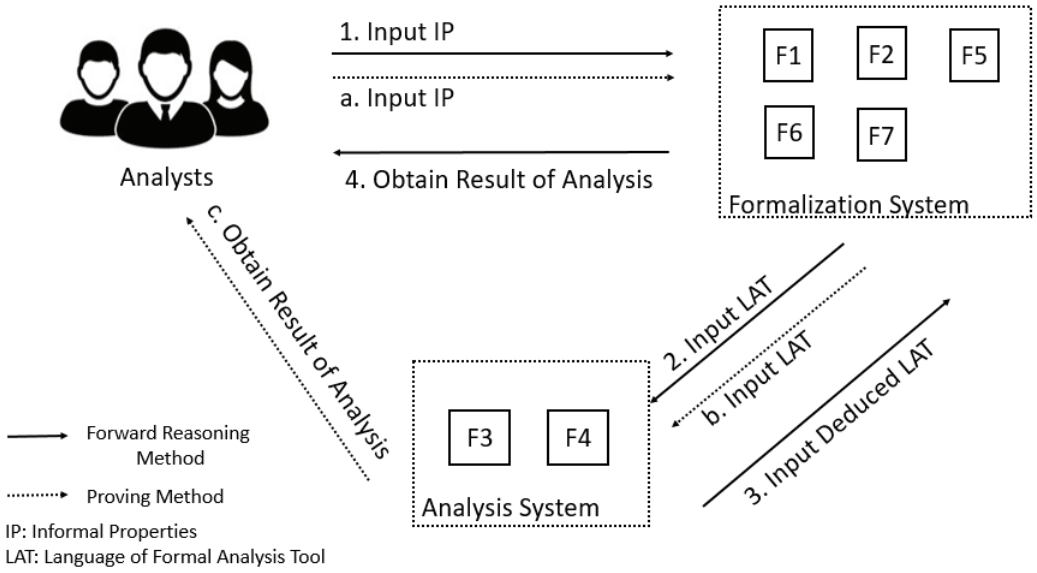

Figure 6.2: Overview of the supporting environement

If analysts would like to use the formal analysis method with reasoning, the properties will be transformed into LF and LAT corresponding to [10] and then forward reasoning is performed by using the result of formalization. After that, the results of forward reasoning presented by LAT will be filtered and sorted. Then the sorted results will be transformed into LF. At last, target LF will be transformed into informal language for analysts to analyze whether there are possible successful attacks or not. If the possible successful attacks are difficult for analysts to confirm, they can use the proving method of the supporting environment to verify whether the possible successful attacks succeed or not. By the two rounds of analysis, analysts can find all flaws in the cryptographic protocols without enumerating the attacks before formal analysis. Analysts can manage the input properties and results of each performing of formal analysis. They can read each property and result of cryptographic protocols by searching the user name or cryptographic name or creation time. Certainly, they can also search, modify or delete them.

Chapter 7

Disscutions

Although we have successfully proved that the extension of formal analysis method with reasoning can find flaws related to confidentiality, authentication, non-repudiation, and fairness, there are still some problems with this method.

The first problem is that only limited types of flaws can be detected. In the method, we use Dolev-Yao model to describe the intruder's abilities and use the confirm behaviors to restrict participant's capabilities. But it cannot describe the capability of all honest participants, dishonest participants and intruders to cross-combine, so the formulas associated with certain flaws may not be deduced.

The second problem is that the method cannot deal with the protocols with $F3$, which is represented as "the participant send data selectively to another". The security of the type of protocols is probabilistic but the method is based on strong relevant logic so that the method cannot detect the probabilistic flaws. In the future, we will consider the degree of the security of protocols and try to solve the problems.

The third problem is that until now, although we have performed different case studies to verify the soundness of the method, there is not a mathematical way to prove that the deduced flaws are really flaws according to the premises. In the future, we will try to prove the soundness of the extended method in a methematical way.

The fourth problem is that the current extended method can detect the flaws as same as proving method, in other words, we haven't found out an unknown flaws in a cryptographic protocol. However, the method does not depend on the analysts' abilities, anyone can detect the flaws by following the steps of the method. In this respect, the extended method is better than the proving method.

The last problem is that a large amount of logical formulas have been deduced by using the extended method, obviously, it is difficult for analysts to analyze them one by one. In the case studies that we have performed, not all the meanings of logical representations have been analyzed. So we need to find some ways to exclude the formulas that considered secure in the generated formulas, narrow the scope of the analysis.

Chapter 8

Conclusions

8.1 Contributions

First, we described the extended formal analysis method with reasoning which can be applied to various cryptographic protocols. As the first step of expansion, we compared 19 representative cryptographic protocols based on participants' number and behaviors and summarized five features. Then, we extended the participants' behaviors corresponding to the summarized features. We also proposed a specific procedure to decided how falsified data that an intruder would send. After extending, we performed case studies to verify the extended method can be used to detect various cryptographic protocols. By succeeding detecting the known flaws of secret splitting protocols, it can be said that the extended method is effective to apply to various cryptographic protocols with four futures.

Second, we proposed the flaw analysis criteria have been proposed. We defined what is a flaw in a cryptographic protocol and considered the security properties of cryptographic protocols can be used as criteria for testing the security of cryptographic protocols. We have organized six fine-grained security properties of confidentiality, authentication, fairness, non-repudiation, anonymity, and atomicity, and used them as flaw analysis criteria to analyze flaws in cryptographic protocols. We also verified the flaw analysis criteria are capable by analyzing Needham Schroeder protocol and anonymous atomic transaction protocols.

Third, to detect more types of cryptographic protocol flaws, we proposed a new method to perform forward reasoning to test the flaws related to non-repudiation and fairness caused by participants' deception. We also took the ISI protocol and CMP1 protocol as cases to perform security analysis. By the result of analyzing, it is proved that the extended formal analysis method with reasoning is effective to detect the flaws related to confidentiality, authentication, non-repudiation and fairness.

Fourth, to resolve the time-consuming and error-prone problems, we explained the first supporting environment for formal analysis of cryptographic protocols. The supporting environment integrates various supporting tools and it can support analysts to perform formal analysis of cryptographic protocols through the whole processes.

Therefore, comparing with the formal analylsis method with reasoning, the extended method is as useful as the proving method and comparing with the previous formal

analysis method with reasoning for key exchange protocols, the extended method is more useful than the previous method.

8.2 Future works

In the future, we will analyze more cryptographic protocols to verify the usefulness of the extended formal analysis method with reasoning for cryptographic protocols. We will also more fully describe the capabilities of intruders and dishonest participants to make it possible to detect flaws related to anonymity and atomicity. We will try to give the methematical proof of the soundness of the extended method. We will find some ways to narrow the scope of the deduced formulas that need to be analyzed and try to find out new flaws of cryptographic protocols.

Publications

Refereed Papers for Doctoral Dissertation

- Jingchen YAN, Kazunori WAGATSUMA, Hongbiao GAO, and Jingde CHENG: A Formal Analysis Method with Reasoning for Cryptographic Protocols, Proc. 12th International Conference on Computational Intelligence and Security, pp. 566-570, Wuxi, China, IEEE Computer Society Press, December 2016.
- Jingchen YAN, Kazunori WAGATSUMA, Hongbiao GAO, and Jingde CHENG: A Supporting Environment for Formal Analysis of Cryptographic Protocols, in J. J. J. H. Park, S.-C. Chen, and K.-K. R. Choo (Eds.), Advanced Multimedia and Ubiquitous Engineering - MUE/FutureTech 2017, Lecture Notes in Electrical Engineering, Vol. 448, pp. 545-550, Springer, Singapore, May 2017.
- Jingchen YAN, Sho ISHIBASHI, Yuichi GOTO, Jingde CHENG: A Study on Fine-Grained Security Properties of Cryptographic Protocols for Formal Analysis Method with Reasoning, Proceedings of 2018 IEEE SmartWorld, Ubiquitous Intelligence and Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People and Smart City Innovations (SmartWorld/UIC/ATC/ ScalCom/CBDCom/IOP/SCI 2018), pp. 210-215, Guangzhou, China, IEEE-CS, Oct 2018.
- Jingchen YAN, Yating WANG, Yuichi GOTO, and Jingde CHENG: An Extension of Formal Analysis Method with Reasoning: A Case Study of Flaw Detection for Non-repudiation and Fairness, 3rd International Conference C2SI-2019 on Codes, Cryptology and Information Security, Lecture Notes in Computer Science (Accepted).

Other Refereed Papers

- Jingchen YAN, Hongbiao GAO, and Jingde CHENG: A Formal Analysis Method with Forward Reasoning for Cryptographic Protocols, Journal of Information Security Research, Vol. 3, No. 5, pp. 462-468, May 2017 (in Chinese).
- Jingchen YAN and Jingde CHENG: A Formal Specification and Verification Method for Security Specification Based on CC, Journal of Information Security Research, Vol. 3, No. 7, pp. 617-623, July 2017 (in Chinese).

- Sho ISHIBASHI, Jingchen YAN, Yuichi GOTO and Jingde CHENG: Primitive Constituent Elements of Cryptographic Protocols, Proceedings of 2018 IEEE SmartWorld, Ubiquitous Intelligence and Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People and Smart City Innovations, pp. 192-197, Guangzhou, China, IEEE Computer Society Press, October 2018.

References

[1] Martin ABADI: Secrecy by typing in security protocols, Journal of the ACM, vol. 46 Issue 5, pp.749-786, September 1999.

[2] Martin Abadi: Security Protocols and their Properties, Bell Labs Research Lucent Technologies, IOS press, 2000

[3] Bruce SCHNEIER: Applied Cryptography: Protocols, Algorithms, and Source Code in C, John Wiley and Sons, Inc, 1996.

[4] Elena ANDREEVA, Andrey BOGDANOV and Bart MENNINK: Towards understanding the known-key security of block ciphers, International Workshop on Fast Software Encryption FSE 2013, Fast Software Encryption, pp 348-366 July, 2014.

[5] Nadarajah ASOKAN: Fairness in Electronic Commerce, Ph.D. thesis, University of Waterloo, Canada, 1998.

[6] Matteo AVALLE, Alfredo PIRONTI and Riccardo SISTO: Formal Verification of Security Protocol Implementations: A Survey, in Formal Aspects of Computing 26.1, pp. 99-123, 2014.

[7] David BASIN, Cas CREMERS and Catherine MEADOWS: Model checking security protocols, Handbook of Model Checking, 2011.

[8] Jason BAU and John C. MITCHELL: Security Modeling and Analysis, IEEE Security and Privacy, Vol. 9, No. 3, pp. 18-25, May-June 2011.

[9] Bruno BLANCHET: An Efficient Cryptographic Protocol Verifier Based on Prolog Rules, Proceedings of the 14th IEEE Computer Security Foundations Workshop, pp. 82-96. IEEE, June 2001.

[10] Bruno BLANCHET: ProVerif 1.90: Automatic Cryptographic Protocol Verifier, User Manual and Tutorial, 2015

[11] Bruno BLANCHET, Ben SMYTH, and Vincent CHEVAL: ProVerif 1.96: Automatic Cryptographic Protocol Verifier, User Manual and Tutorial, 2016

[12] Colin BOYD and Anish MATHURIA: Protocols for Authentication and Key Establishment, Springer, June 2003.

[13] Schneier BRUCE: Applied Cryptography: Protocols, Algorithms, and Source Code in C, John Wiley and Sons, Inc, 1996.

[14] Bruno BLANCHET: Automatic proof of strong secrecy of security protocols, IEEE Symposium on Security and Privacy, 2004.

[15] Andrew BUTTERFIELD and George NGONDI: Oxford dictionary of Computer Science, Oxford University Press, 2016.

[16] Jean CAMP, Michael HARKAVY, D TYGAR and Benjamin YEE: Anonymous Atomic Transaction, Proc. of the 2nd USENIX workshop on Electronic Commerce, pp. 123-133, 1996

[17] Jingde CHENG: A Strong Relevant Logic Model of Epistemic Processes in Scientific Discovery, Information modeling and knowledge bases XI, Vol. 61, pp. 136-159, February 2000.

[18] Jingde CHENG: Strong Relevant Logic as the Universal Basis of Various Applied Logics for Knowledge Representation and Reasoning, Frontiers in Artificial Intelligence and Applications, Vol. 136, pp. 310-320, February 2006.

[19] Jingde CHENG and Junichi MIURA: Deontic Relevant Logic as the Logical Basis for Specifying, Verifying, and Reasoning about Information Security and Information Assurance, Proceedings of the 1st International Conference on Availability, Reliability and Security, pp. 601-608, IEEE-CS, April 2006.

[20] Jingde CHENG, Shinsuke NARA, and Yuichi GOTO: FreeEnCal: A Forward Reasoning Engine with General-Purpose, Knowledge-Based Intelligent Information and Engineering Systems, Lecture Notes in Artificial Intelligence, Vol. 4693, pp. 444-452, Springer-Verlag, September 2007.

[21] Edmund CLARKE, Orna GRUMBERG, Daniel KROENING and Helmut VEITH: Model Checking. Cyber-Physical Systems, 2018.

[22] John Clark and Jeremy JACOB: A survey of authentication protocol literature, http://www.cs.youk.ac.uk.jac/, 2006

[23] Veronique CORTIER, Steve KREMER and Bogdan WARINSCHI: A survey of symbolic methods in computational analysis of cryptographic systems. Journal of Automated Reasoning 46(3-4) 225-259, 2011

[24] Cas CREMERS: The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols, Computer Aided Verification, Lecture Notes in Computer Science, Vol. 5123, pp. 414-418, Springer, July 2008.

[25] Cas CREMERS: Scyther User Manual, 2014

[26] Robert H. DENG, Li GONG, Aurel A. LAZER, Weiguo WANG: Practical protocols for certified electronic mail, Journal of Network and Systems Management, Vol. 4 (3), pp. 279297, 1996

[27] Dorothy DENNING and Giovanni SACCO: Timestamps in key distribution protocols, Communications of the ACM, vol. 24 (8), pp. 198-208, 1981

[28] Whitfield DIFFIE and Martin HELLMAN: New directions in cryptography, IEEE Transactions on Information Theory Society, pp. 644-654, 1976.

[29] Danny DOLEV and Andrew YAO: On the Security of Public Key Protocols, IEEE Transactions on Information Theory, Vol. 29, No. 2, pp. 198-208, IEEE, March 1983.

[30] Maria FASLI: Agent technology for e-commerce. Chichester: John Wiley and Sons, 2007.

[31] Kokichi FUTATSUGI and Razvan DIACONESCUD, CafeOBJ report, in World Scientific, 1998.

[32] Ralf HAUSER, Michael STEINER and Michael WAIDNER: Micro-payments based on IKP. IBM Zurich Research Laboratory, IBM Research Division Report RZ279, Zurich, Switzerland, 1996

[33] George HEIDORN: Automatic Programming Through Natural Language Dialogue: A Survey. In: Readings in Artificial Intelligence and Software Engineering, pp. 203–214, 1986

[34] Sho ISHIBASHI, Jingchen YAN, Yuichi GOTO and Jingde CHENG: Primitive Constituent Elements of Cryptographic Protocols, Proceedings of 2018 IEEE SmartWorld, Ubiquitous Intelligence and Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People and Smart City Innovations (SmartWorld/UIC/ATC/ScalCom/CBDCom/IOP/SCI 2018), pp. 192-197, Guangzhou, China, IEEE-CS, Oct 2018

[35] International Organization for Standardization. ISO/IEC 13888-3: Information Security techniques - non-repudiation - Part: Mechanisma using asymmetric techniques, 1997

[36] International Organization for Standardization. ISO/IEC 13888-3: Information Security techniques - non-repudiation - Part: Mechanisma using symmetric techniques, 1998

[37] International Organization for Standardization. ISO/IEC 29128: Information technology - Security techniques - Verification of cryptographic protocols, 2011

[38] Steve KREMER, Olivier MARKKOWITCH and Jianying ZHOU: An intensive survey of fair non-repudiation protocols. Computer Communications 25 (17), 1606-1621, 2002

[39] Gavin LOWE: Analyzing protocols subject to guessing attacks, Journal of Computer Security vol.12 (1), pp. 83-89, 2004

[40] Gennady MEDVINSKY and Clifford NEUMAN: Netcash: a design of practical electronic currency on the internet. In: 1st ACM conference on Computer and communications security, pp.102-106. Fairfax, Virginia, USA, 1993

[41] Catherine MEADOWS: Formal verification of cryptographic protocols: a survey. In: International Conference on the Theory and Application of Cryptology, Springer Berlin Heidelberg, 1994

[42] Alfred MENEZES, Paul OORSCHOT, and Scott VANSTONE: Handbook of applied cryptography, CRC Press, 1997.

[43] Catherine MEADOWS: Formal methods for cryptographic protocol analysis: emerging issues and trends. IEEE Journal on selected areas in communications 21 (1), 44-54, 2003

[44] Roger NEEDHAM and Michael SHROEDER: Using encryption for authentication in large networks of computers, Communications of the ACM, vol. 21 (12), ACM, pp.993-999, 1978

[45] National Institute of Information and Communications Technology, Cryptographic Protocol Verification Portal, http://crypto-protocol.nict.go.jp/, accessed March 9, 2015.

[46] Dave OTWAY and Owen REES: Efficient and Timely Mutual Authentication, ACM SIGOPS Operating Systems Review, vol. 21 (1), pp. 8-10, ACM, January 1987.

[47] Markowitch OLIVIER, Dieter GOLLMANN and Steve KREMER: On fairness in exchange protocols. In: Proceedings of 2002 International Conference on information Security and cryptography, LNCS, vol. 2587, pp. 451-464. Seoul, Korea 2002

[48] Andreas PFITZMANN and Michael WAIDNER: Networks without user observability, Computers and Security vol. 6 (2), pp. 158-166, April 1987.

[49] Lawrence PAULSON: Proving Properties of Security Protocols by Induction, Proceedings of 10th IEEE Computer Security Foundations Workshop, pp. 70-83, IEEE, June 1997.

[50] Lawrence PAULSON: The Inductive Approach to Verifying Cryptographic Protocols, Journal of Computer Security, Vol. 6, No. 1-2, pp. 85-128, February 1998.

[51] Eric RESCORLA and Brian KORVER: RFC 3552: Guidelines for Writing RFC Text on Security Considerations, July 2003.

[52] Michael ROE: Cryptography and Evidence. Ph.D. thesis,Computer Laboratory, University of Cambridge, 1997

[53] Paul SYVERSON: A taxonomy of replay attacks. Proceedings The Computer Security Foundations Workshop VII,pp,187-191.IEEE,14-16 June 1994

[54] Doug TYGAR: Atomicity in electronic commerce, Proc. PODC '96 Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing, pp. 8-26, Philadelphia, Pennsylvania, May, 1996.

[55] Kazunori WAGATSUMA, Yuichi GOTO, and Jingde CHENG: A Formal Analysis Method with Reasoning for Key Exchange Protocols, IPSJ Journal, Vol. 56, No. 3, pp. 903-910, IPSJ, March 2015 (in Japanese).

[56] Kazunori WAGATSUMA, Tsubasa HARADA, Shogo ANZE, Yuichi GOTO, and Jingde CHENG: A Supporting Tool for Spiral Model of Cryptographic Protocol Design with Reasoning-Based Formal Analysis, Advanced Multimedia and Ubiquitous Engineering - Future Information Technology, Lecture Notes in Electrical Engineering, Vol. 354, pp. 25-32, Springer, Heidelberg, July 2015.

[57] Jingchen YAN, Kazunori WAGATSUMA, Hongbiao GAO, and Jingde CHENG: A Formal Analysis Method with Reasoning for Cryptographic Protocols, Proc. 12th International Conference on Computational Intelligence and Security, pp. 566-570, Wuxi, China, IEEE Computer Society, December 2016.

[58] Jianying ZHOU and Dieter GOLLMANN: Evidence and non-repudiation. Journal of Network and Computer Applications, 20(30), 267-281, 1997

Appendix A

Case Study In Secret Splitting Protocol

A.1 Logic Formulas of Secret Splitting Protocol

A.1.1 First Protocol

- $Start(p_1, p_2) \Rightarrow Send(p_1, s, data(id(p_1), id(p_2)))$
- $Recv(s, data(id(p_1), id(p_2))) \Rightarrow (Parti(s) \Rightarrow Send(s, p_1, enc(symk(p_1, s), r_1)))$
- $Recv(s, enc(symk(p_1, s), r_1)) \Rightarrow (Parti(s) \Rightarrow Send(s, p_2, enc(symk(p_2, s), r_2)))$
- $Start(p_1, p_2) \Rightarrow Get(i, data(id(p_1), id(p_2)))$
- $Start(p_1, p_2) \Rightarrow Get(i, enc(symk(p_1, s), r_1))$
- $Start(p_1, p_2) \Rightarrow Get(i, enc(symk(p_2, s), r_2))$
- $Get(p_1, enc(symk(p_1, p_2), x)) \Rightarrow Get(p_1, x)$
- $Get(p_1, enc(symk(p_2, p_1), x)) \Rightarrow Get(p_1, x)$
- $Parti(a)$
- $Parti(b)$
- $Parti(s)$
- $\neg Parti(i)$
- $Sesk(sesk)$

- $Sesk(old(sesk))$
- $Recv(p, x) \Rightarrow Get(p, x)$
- $Send(i, p, x) \Rightarrow Recv(p, x)$
- $Send(p_1, p_2, x) \Rightarrow Get(i, x)$
- $Send(p, i, x) \Rightarrow Recv(i, x)$
- $Start(a, b)$

A.1.2 Second Protocol

- $Start(p_1, p_2) \Rightarrow Send(s, p_1, enc(symk(p_1, s), r_1, id(p_2)))$
- $Recv(s, enc(symk(p_1, s), r_1, id(p_2))) \Rightarrow (Parti(s) \Rightarrow Send(s, p_2, enc(symk(p_2, s), r_2, id(p_1))))$
- $Start(p_1, p_2) \Rightarrow Get(i, enc(symk(p_1, s), r1, id(p_2)))$
- $Start(p_1, p_2) \Rightarrow Get(i, enc(symk(p_2, s), r_2, id(p_1)))$
- $Get(p_1, enc(symk(p_1, p_2), x_1, x_2)) \Rightarrow Get(p_1, data(x_1, x_2))$
- $Get(p_1, enc(symk(p_2, p_1), x_1, x_2)) \Rightarrow Get(p_1, data(x_1, x_2))$
- $Parti(a)$
- $Parti(b)$
- $Parti(s)$
- $\neg Parti(i)$
- $Sesk(sesk)$
- $Sesk(old(sesk))$
- $Recv(p, x) \Rightarrow Get(p, x)$

- $Send(i, p, x) \Rightarrow Recv(p, x))$
- $Send(p_1, p_2, x) \Rightarrow Get(i, x)$
- $Send(p, i, x) \Rightarrow Recv(i, x)$
- $Start(a, b)$

A.1.3 Third Protocol

- $Start(p_1, p_2) \Rightarrow Send(p_1, p_2, data(id(p_1), nonce(p_1)))$
- $Recv(p_2, data(id(p_1), nonce(p_1)))$
  $\Rightarrow (Parti(p_2) \Rightarrow Send(p_2, s, data(id(p_1), nonce(p_1), id(p_2), nonce(p_2))))$
- $Recv(s, data(id(p_1), nonce(p_1), id(p_2), nonce(p_2)))$
  $\Rightarrow (Parti(s) \Rightarrow Send(s, p_1, enc(symk(p_1, s), r_1, id(p_2), nonce(p_1))))$
- $Recv(s, enc(symk(p_1, s), r_1, id(p_2), nonce(p_1)))$
  $\Rightarrow (Parti(s) \Rightarrow Send(s, p_2, enc(symk(p_2, s), r_2, id(p_1), nonce(p_2))))$
- $Start(p_1, p_2) \Rightarrow Get(i, data(id(p_1), old(nonce(p_1))))$
- $Start(p_1, p_2) \Rightarrow Get(i, data(id(p_1), old(nonce(p_1)), id(p_2), old(nonce(p_2))))$
- $Start(p_1, p_2) \Rightarrow Get(i, enc(symk(p_1, s), r_1, id(p_2), old(nonce(p_1))))$
- $Start(p_1, p_2) \Rightarrow Get(i, enc(symk(p_2, s), r_2, id(p_1), old(nonce(p_2))))$
- $Get(p_1, enc(symk(p1, p2), x_1, x_2, x_3)) \Rightarrow Get(p_1, data(x_1, x_2, x_3))$
- $Get(p_1, enc(symk(p_2, p_1), x_1, x_2, x_3)) \Rightarrow Get(p_1, data(x_1, x_2, x_3))$
- $Parti(a)$
- $Parti(b)$
- $Parti(s)$
- $Parti(i)$

- $Sesk(sesk)$
- $Sesk(old(sesk))$
- $Recv(p, x) \Rightarrow Get(p, x)$
- $Send(i, p, x) \Rightarrow Recv(p, x)$
- $Send(p_1, p_2, x) \Rightarrow Get(i, x)$
- $Send(p, i, x) \Rightarrow Recv(i, x)$
- $Start(a, b)$

A.2 Symbols of FreeEnCal

Individual constants

- C_0_0: a
- C_0_1: b
- C_0_2: s
- C_0_3: i
- C_0_15: $secret1$
- C_0_16: $secret2$

Functions

- C_0_5: $symk(p1, p2)$
- C_0_8: $id(p)$
- C_0_9: $nonce(p)$
- C_0_10: $tstamp(p)$
- C_0_11: $enc(k, x_1, ..., x_n)$

- C_0_12: $data(x_1, ..., x_n)$
- C_0_14: $old(x)$

Predicates

- $C\_1\_0 : Send(p1, p2, x)$
- $C\_1\_1 : Recv(p.x)$
- $C\_1\_2 : Get(p, x)$
- $C\_1\_3 : Start(p1, p2)$
- $C\_1\_4 : Parti(p)$
-

Individual variables

- $V\_0\_0 : p1$
- $V\_0\_1 : p2$
- $V\_0\_2 : s$

Logical connectives

- C_2_0: $\Rightarrow$
- C_2_1: $\wedge$
- C_2_2: $\neg$

Appendix B

Case Study In ISI Protocol

B.1 Logic Formulas of ISI Protocol

- $Start(a, b)$
- $Start(p_1, p_2) \Rightarrow Send(p_1, p_2, data(symk(p_1, p_2)))$
- $Recv(p_2, data(symk(p_1, p_2))) \Rightarrow (Parti(p_2) \Rightarrow Send(p_2, p_1, data(enc(symk(p_1, p_2), pk(p_2)))))$
- $Recv(p_1, data(enc(symk(p_1, p_2), pk(p_2))))$
  $\Rightarrow (Parti(p_1) \Rightarrow Send(p_1, p_2, data(enc(pk(p_2), sig(cs, coins), sk(p_1)))))$
- $Recv(p_2, data(enc(pk(p_2), sig(cs, coins), sk(p_1))))$
  $\Rightarrow (Parti(p_2) \Rightarrow Send(p_2, cs, data(enc(pk(cs), sig(cs, coins), sk(p_2)))))$
- $Recv(cs, data(enc(pk(cs), sig(cs, coins), sk(p_2))))$
  $\Rightarrow (Parti(cs) \Rightarrow Send(cs, p_2, data(enc(sk(p_2), sig(cs, new - coins)))))$
- $Recv(p_2, data(enc(pk(p_2), sig(cs, new - coins))))$
  $\Rightarrow (Parti(cs) \Rightarrow Send(p_2, p_1, data(enc(sk(p_1), sig(p_2, amount)))))$
- $Parti(a)$
- $Parti(b)$
- $Send(a, b, data(x)) \Rightarrow Recv(b, data(x))$
- $Recv(a, \{x\}_{k_{ab}}) \Rightarrow Recv(a, data(x))$
- $Recv(a, \{x\}_{k_{ba}}) \Rightarrow Recv(a, data(x))$

- $Recv(a, \{x\}_{pk_a}) \Rightarrow Recv(a, data(x))$
- $Recv(a, \{x\}_{sk_a}) \Rightarrow Recv(a, data(x))$
- $Recv(a, data(x_1, x_2)) \Rightarrow Recv(a, data(x_1))$
- $Recv(a, data(x_1, x_2)) \Rightarrow Recv(a, data(x_2))$
- $cof(a, cs, pk(cs))$
- $cof(b, cs, pk(cs))$
- $Recv(P, data(sig(q, x))) \wedge cof(P, Q, kq) \Rightarrow cof(P, Q, x)$
- $cof(P, Q, data(X, Y, Z)) \Rightarrow cof(P, Q, X)$
- $cof(P, Q, data(X, Y, Z)) \Rightarrow cof(P, Q, Y)$
- $cof(P, Q, data(X, Y, Z)) \Rightarrow cof(P, Q, Z)$
- $cof(P, TTP, m) \Rightarrow cof(P, Q, m)$
- $cof(P, Q, \{m\}_k) \wedge cof(P, Q, k) \Rightarrow cof(P, Q, m)$
- $cof(P, Q, sig(q, m)) \wedge cof(P, Q, kq) \Rightarrow cof(P, Q, m)$
- $cof(a, cs, pk(cs))$
- $cof(b, cs, pk(cs))$

B.2 Symbols of FreeEnCal

Individual constants

- C_0_0: a
- C_0_1: b
- C_0_2: cs

- C_0_3: i
- C_0_4: $sesk$
- C_0_15: $coins$
- C_0_17: $amount$
- C_0_18: $new - coins$
- C_0_19: $amount'$

Functions

- C_0_5: $symk(p1, p2)$
- C_0_6: $pk(p)$
- C_0_7: $sk(p)$
- C_0_8: $id(p)$
- C_0_9: $nonce(p)$
- C_0_10: $tstamp(p)$
- C_0_11: $enc(k, x1, ..., xn)$
- C_0_12: $date(x1, ..., xn)$
- C_0_13: $plus(p)$
- C_0_14: $old(x)$
- C_0_16: $sig(p, x)$

Predicates

- $C\_1\_0 : Send(p1, p2, x)$
- $C\_1\_1 : Recv(p.x)$

- $C\_1\_2 : Get(p, x)$
- $C\_1\_3 : Start(p1, p2)$
- $C\_1\_4 : Parti(p)$
- $C\_1\_5 : Sesk(p)$
- $C\_1\_6 : cof(p, q, x)$
- $C\_1\_7 : sig(p, x)$

Individual variables

- $V\_0\_0 : p1$
- $V\_0\_1 : p2$
- $V\_0\_2 : CS$
- $V\_0\_3 : Y$
- $V\_0\_4 : Z$
- $V\_0\_5 : P$
- $V\_0\_6 : Q$
- $V\_0\_7 : m$
- $V\_0\_8 : coins$
- $V\_0\_9 : new - coins$
- $V\_0\_10 : amount$
- $V\_0\_11 : X$

Logical connectives

- C_2_0: $\Rightarrow$
- C_2_1: $\wedge$
- C_2_2: $\neg$