Doctoral Dissertation

A Formal Analysis Method with Reasoning for Cryptographic Protocols and Its Supporting Tools

暗号プロトコルに関する推論的形式分析手法とその支援ツール

Kazunori Wagatsuma

Graduate School of Science and Engineering, Saitama University

Supervisor: Professor Jingde Cheng

March 2016

Abstract

Many cryptographic protocols have been proposed to securely send and receive information among agents in unsecured networks for various purposes. However, an intruder can eavesdrop secret data and/or impersonate in unsecured cryptographic protocols. Therefore, formal analysis methods of cryptographic protocols such as model checking and theorem proving are used to try to find flaws in the protocols. In formal analysis method such as theorem proving and model checking, analysts firstly enumerate fatal actions of attacks before doing analysis, and then verify whether those attacks succeed or not in communication processes of that target cryptographic protocols. I can say that model checking and theorem proving are formal analysis method with proving because analysts must enumerate target of verification before doing analysis in those methods. In formal analysis method with proving, analysts must enumerate all fatal actions of attacks for target protocol before doing analysis when they verify that target cryptographic protocol is secure. In other words, analysts cannot verify those attacks that are not enumerated before analysis. As an alternative way, a concept of formal analysis with reasoning for cryptographic protocols has been proposed that deduce actions of participants and an intruder from behaviors explicitly and implicitly included in specifications of cryptographic protocols by forward reasoning and detect fatal actions of attacks without enumerated those fatal actions before analysis. Formal analysis method with reasoning is included 3 phases, formalizing specification of a cryptographic protocol, forward reasoning from result of formalization as premises, and analyzing deduced logical formulas by forward reasoning. However, it is difficult for analysts to perform formal analysis with reasoning because its concrete and general steps are not established.

This paper presents a formal analysis method with reasoning for cryptographic protocols and its supporting tools. At first, I proposed the concrete and general method of formalization, forward reasoning, and analysis in formal analysis method with reasoning. After that, I showed that the proposed method is effective for formal analysis of cryptographic protocols. Finally, I implemented supporting tools in order to perform the proposed method automatically, and evaluated those supporting tools.

I proposed concrete and general steps of formal analysis method with reasoning for key exchange protocols, and extended the proposed method in order to be applied to various cryptographic protocols. In order to extend the proposed method, at first, I compared 19 kinds of cryptographic protocols based on participants' behaviors and the number of participants from web site "Cryptographic Protocol Verification Portal," and a book "Applied Cryptography: Protocols, Algorithms, and Source Code in C" and I found five differences among cryptographic protocols. After that, I extended the proposed method in order to represent those five differences. In order to show effectiveness of the proposed method, at first, I re-detected fatal actions of attacks in target cryptographic protocols that are pointed out by using the proposed method. After that, I chose some cryptographic protocols that include those five differences and performed formal analysis of those cryptographic protocols by the proposed method. As the result, I could perform tasks of the proposed method to the end. Therefore, it can be said that the proposed method can apply to detect fatal actions of attacks that represent success of attack in 19 cryptographic protocols. In order to support tasks of the proposed method, I implemented supporting tools that are inputted specification of target cryptographic protocol, and perform tasks of the proposed method automatically, and output the result whether attacks succeed or not. Finally, I evaluated whether duration in the tasks can be reduced by using those supporting tools, compared to a case that analysts perform the proposed method manually.

As the result of showing effectiveness of the proposed method, at first, I could re-detect fatal actions of attacks that represent success of attack that is pointed out. Therefore, I showed that the proposed method can detect fatal actions of attacks that represent success of attack. After that, in formal analysis of some cryptographic protocols that include five differences among cryptographic protocols, I could perform tasks of the proposed method to the end. Therefore, I showed that the proposed method can be applied to those 19 cryptographic protocols. As the result of evaluation of implemented supporting tools, I showed that analysts can reduce duration of the proposed method by using those supporting tools.

Structure of this thesis is as follows. Chapter 1 presents background, motivation, and purpose of this research. Chapter 2 explains formal analysis of cryptographic protocols. Chapter 3 proposes formal analysis method with reasoning for key exchange protocols. Chapter 4 presents extending the proposed formal analysis method with reasoning for key exchange protocols. Chapter 5 shows effectiveness of the proposed method. Chapter 6 presents supporting tools for the proposed method and evaluation of those supporting tools. Chapter 7 presents discussion of the proposed method and implemented supporting tools. Finally, concluding remarks are given in Chapter 8.

Acknowledgments

I would like to express my special thanks to my thesis supervisor, Professor Jingde Cheng for his enthusiastic guidance, understanding, and invaluable support on all aspects of academic life to help me accomplish my thesis and pursue my doctoral degree.

I am very grateful to my thesis committee: Professor Norihiko Yoshida, Professor Atsushi Uchida, Associate Professor Noriaki Yoshiura, and Associate Professor Yuichi Goto for their support, valuable feedback, and insightful ideas to this research.

I am also grateful to Associate Professor Yuichi Goto, and Assistant Professor Hongbiao Gao for teaching me how to do research and helping me in all respect through the hard moments of graduate school. I am also grateful to other AISE lab members to give me advice and help in PhD research.

I would also like to express my special thanks to my family and friends for their unfailing support through the hard moments of my graduate studies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

Many cryptographic protocols have been proposed to securely send and receive information among agents in unsecured networks for various purposes, for example, key exchange, authentication, digital signature, secret splitting, e-voting, zero-knowledge proof, and so on [31]. In particular, many key exchange protocols have been proposed and used [5] [10].

A cryptographic protocol is not secure if, at least, an attack by an intruder is succeeded in the communication process of that protocol. In a communication process of a cryptographic protocol, an attack is a sequence of actions by participants and an intruder. Success of an attack is that an intruder can do the last action of the attack. A cryptographic protocol has a flaw if success of an attack can occur in the communication process of that protocol. In this paper, I call the last action of an attack 'a fatal action of an attack.' A behavior is a set of rules among events and/or actions in communication processes of cryptographic protocols. The rule can be represented as 'if a certain event occurs, then an agent does a certain action' or 'if an agent does a certain action, the agent does an other action.

Formal analysis of cryptographic protocols is used to try to find flaws in the protocols [31]. While many cryptographic protocols have been proposed, knowledge and technique of an intruder have developed day after day. Therefore, flaws of proposed protocols often found and attacked after those protocols had been used. In those unsecured cryptographic protocols, an intruder can eavesdrop secret data and/or impersonate other participate. If flaws of proposed cryptographic protocols are found, it is time-consuming to improve those cryptographic protocols. Furthermore, human life, nations, and money are severely affected if flaws are found and attacked in cryptographic protocols that are used for military, medical, and politic fields. Therefore, flaws must not exist in cryptographic protocols that are used for those fields.

Model checking and theorem proving are used as formal analysis methods of cryptographic protocols [3]. In those methods, analysts firstly enumerate fatal actions of attacks, and then verify whether those fatal actions are performed or not in communication processes of the target cryptographic protocols. The process of the method is proving because analysts must enumerate target of verification before doing analysis in those methods. In other words, analysts cannot verify those fatal actions of attacks that are not enumerated before analysis.

As an alternative way, a concept of formal analysis with reasoning for cryptographic protocols has been proposed [8]. In the analysis, forward reasoning is used to deduce actions of participants and an intruder from behaviors explicitly and implicitly included in specifications of cryptographic protocols by forward reasoning, and detect fatal actions of attacks without enumerated before analysis. Formal analysis method with reasoning is included 3 phases, formalizing specification of a cryptographic protocol, forward reasoning from result of formalization as premises, and analyzing deduced logical formulas by forward reasoning [8]. However, it is difficult for analysts to perform formal analysis with reasoning because its concrete and general steps are not established.

1.2 Purposes and Objectives

This paper presents a formal analysis method with reasoning for cryptographic protocols and its supporting tools. At first, I propose the concrete and general method of formalization, forward reasoning, and analysis in formal analysis method with reasoning. After that, I show that the proposed method is effective for formal analysis of cryptographic protocols. Finally, I implement supporting tools in order to easily perform the proposed method for analysts, and evaluate those supporting tools.

1.3 Structure of This Thesis

The rest of this paper is organized as follows. Chapter 2 explains formal analysis of cryptographic protocols. Chapter 3 proposes formal analysis method with reasoning for key exchange protocols. Chapter 4 presents extending the proposed formal analysis method with reasoning for key exchange protocols. Chapter 5 shows effectiveness of the proposed method. Chapter 6 presents supporting tools for the proposed method and evaluation of those supporting tools. Chapter 7 presents discussion of the proposed method and implemented supporting tools. Finally, concluding remarks are given in Chapter 8.

Chapter 2

Formal Analysis of Cryptographic Protocols

2.1 Cryptographic Protocols

Protocols are steps for achieving a certain purpose that is related to two or more participants [31]. Cryptographic protocols are protocol using cryptography. Therefore, cryptographic protocols are steps for achieving a certain purpose that is related to two or more participants using cryptography.

It is different for each cryptographic protocol what purpose and what data is sent and received. For example, there are following kinds of cryptographic protocols [31].

- Key exchange

  Participants exchange a session key for securely sending and receiving data in unsecured network.

- Authentication

  Participants prove that a target of sending and receiving data is correct.

- Secret splitting

  Participants split secret data into several parts of data, and prevent restoring secret data without collecting each split data.

- Digital signature

  Participants sign in unsecured network with preventing attack of an intruder, for example, copy and falsifying of signature, spoofing.

- Timestamp

  Participants add timestamp in order to prove that there was a document at specific time with preventing attack of an intruder, for example, copy and falsifying of timestamp or the document.

- Zero knowledge proofs

  A participant proves that the participant knows an information without knowing the information to another participant.

In unsecured network, an intruder can various attack, for example, eavesdropping secret data and/or impersonating other participate. Cryptographic protocols are effective for preventing such attacks by an intruder, and participants can securely send and receive data in unsecured network.

2.2 Formal Analysis Method with Proving

Model checking and theorem proving are used as formal analysis methods of cryptographic protocols. Although Protocol Composition Logic (PCL) [12] and pi-calculus [1] are also used for formal analysis of cryptographic protocols, both methods can be performed by model checking systems [4], [11].

Model checking is a verification method that inputs formalized fatal actions of attacks, participants' and an intruder's behavior that are represented by finite-state in order to explore whether there are reachable states that those attacks succeed or not in participants' and the intruder's behavior [30]. If there are states that represent those fatal actions perform, it can be said that attacks succeed in the target cryptographic protocol. Typical model-checking methods and tools are Scyther [11] and ProVerif [4].

Theorem proving is a verification method that inputs fatal actions of attacks, participants' and an intruder's behavior formalized by logical formulas in order to prove whether those attacks succeed or not in the target cryptographic protocol [30]. If it is proved that those fatal actions are performed in participants' and the intruder's behavior as premises, it can be said that attacks succeed in the target cryptographic protocol. Typical theorem-proving tools are Isabelle [28] and CafeOBJ [14].

Formal analysis methods such as model checking and theorem proving are based on proving. Formal analysis method with proving is a method that analysts firstly enumerate fatal actions of attacks before doing analysis, and then verify whether those attacks succeed or not in the communication process of that target cryptographic protocol. Overview of formal analysis method with proving is showed in figure 2.1. At first, analysts formalize participants' and an intruder's behavior, and enumerated fatal actions of attacks as premises. After that, the analysts verify whether those fatal actions are performed in participants' and the intruder's behaviors. If it is verified that those fatal actions are performed, it can be said that those attacks succeed in the target cryptographic protocol. The process of the method is proving because analysts must previously enumerate target of verification in those methods. Model checking and theorem proving are used to automatically perform analysis whether those attacks succeed in the formalized protocol or not [3]. Therefore, formal analysis methods such as model checking and theorem proving are based on proving.

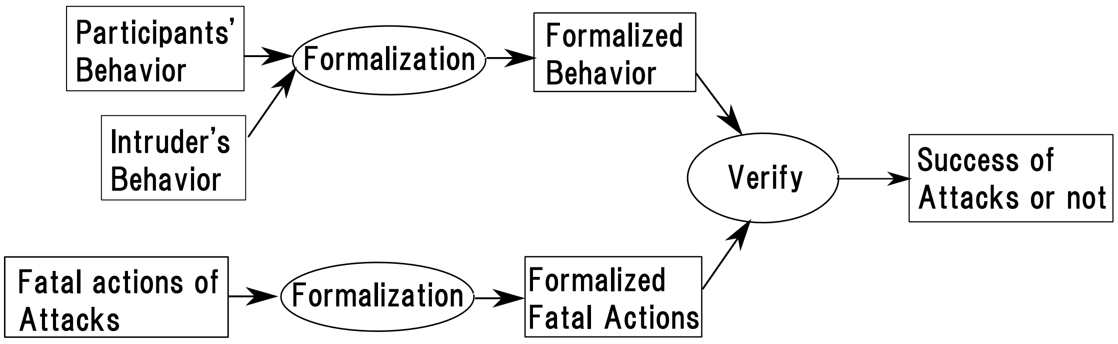


Figure 2.1: Overview of formal analysis method with proving.

Formal analysis methods with proving processes have a limitation. A cryptographic protocol is not secure if, at least, an attack by an intruder is succeeded in the communication process of that protocol. Therefore, in formal analysis method with proving such as model checking and theorem proving, analysts must enumerate all fatal actions of attacks for target protocol before doing analysis when they verify that the target cryptographic protocol is secure, because analysts cannot verify those fatal actions of attacks that are not enumerated before analysis. However, it is difficult for analysts to enumerate all fatal actions of attacks.

2.3 Formal Analysis Method with Reasoning

As an alternative way, a concept of formal analysis with reasoning for cryptographic protocols has been proposed [8]. Overview of proposed formal analysis method with reasoning is showed in figure 2.2. In the proposed method, analysts formalize participants' and an intruder's behaviors of target protocol, and deduce actions of participants and an intruder from behaviors explicitly and implicitly included in specifications of cryptographic protocols. If fatal actions of attacks are detected from deduced actions, those attacks succeed in the target cryptographic protocols.

Analysts do not need to enumerate fatal actions of attacks by an intruder before doing the analysis in formal analysis method with reasoning because actions of participants and an intruder are deduced by forward reasoning from participants' and an intruder's behaviors as premises, and those fatal actions of attacks are detected. Therefore, formal analysis method with reasoning can support to find fatal actions of attacks that analysts did not enumerate before doing analysis.

In the method, strong relevant logics [6], [7] are appropriate for logic systems underlying forward reasoning because in any reasoning based on strong relevant logics those conclusions that are not related to premises are not deduced. Furthermore, as many logical formulas should be deduced from formalized specification of cryptographic protocol by forward reasoning [20]. Therefore, forward reasoning engine FreeEnCal that can automatically perform forward reasoning has been proposed and developed [9]. FreeEnCal can handle any logic system and formal theory by input vocabulary, configuration rules, axiom, and inference rule of logical formulas. Therefore, logical formulas

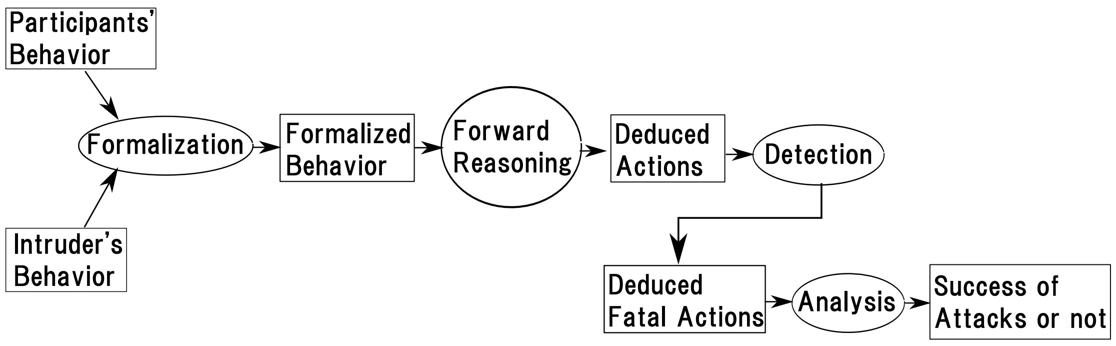


Figure 2.2: Overview of formal analysis method with reasoning.

can be automatically deduced from inputted logical formulas as premises. However, it is difficult for analysts to perform formal analysis with reasoning because its concrete and general steps are not established.

Chapter 3

Formal Analysis Method with Reasoning for Key Exchange Protocols

3.1 Key Exchange Protocols

Key exchange protocols are steps for securely sending and receiving data in unsecured network to exchange a session key between two participants. Participants can send and receive any data by using exchanged session key in only particular session. Key exchange protocols can be classified into 3 types [5].

- Server-based key exchange protocol using symmetric key
- Server-less key exchange protocol using symmetric key
- Server-less key exchange protocol using public key

Participants send and receive data by using their encrypt key that is defined in each 3 types of key exchange protocols for exchanging a session key. In key exchange protocols, eavesdropping and falsification the session key by an intruder must not be possible.

Generally, in specification of cryptographic protocols, sending and receiving in step M are represented as eq.(3.1). It means that X_1 sends data $Y_1, Y_2, \cdots, Y_z$ ($z \in \mathbb{N}$) to X_2 in step M.

$$M.X_1 \rightarrow X_2 : Y_1, Y_2, \cdots, Y_z \tag{3.1}$$

Persons and a trusted server are substituted in X_1 and X_2. Persons are defined as A, B, and a trusted server is defined as S. In the method, persons and a trusted server are called participants. Any kinds of data are substituted in Y_i ($i \in \mathbb{N}, 1 \leq i \leq z$). $\{Y_1, Y_2, \cdots, Y_z\}_K$ ($z \in \mathbb{N}$) means that encrypted data by key K. Following data are mainly used in many key exchange protocols.

- A, B: Identifiers of A, B.
- S: Identifiers of a trusted server.
- I: Identifier of an intruder.

- $K_{X_1X_2}$: Symmetric key of participants X_1 and X_2.
- E_X: Public key of a participant X.
- Sig_X: Signature of a participant X.
- N_X: Nonce of a participant X.
- T_X: Timestamp of X.
- $\{Y_1, Y_2, \cdots, Y_z\}_K$: Encrypted $Y_1, Y_2, \cdots, Y_z$ by key K.
- $Sig_X(Y)$: Data with signature of X.
- $Y + 1$: Incremented data of Y.

In addition, there are uniquely defined data in each protocol.

Many key exchange protocols have been proposed and used among cryptographic protocols [5], [10]. Key exchange is a fundamental method for participants to securely send and receive data in unsecured network [5]. On the other hand, formal analysis of 59 cryptographic protocols that are recently proposed has been performed based on an international standard ISO/IEC 29128 [22], and 56 of them are key exchange protocols [26]. Therefore, key exchange protocols are one of the most important method among cryptographic protocols.

3.2 Tasks of Formalization

3.2.1 Overview of Formalization

I investigated behavior of participants in proposed key exchange protocols [5], [31]. As the result, I found that participants repeat following actions.

1. A participant sends any data to another participant.
2. Receiver receives the data.
3. If the receiver receives encrypted data, and has corresponding decryption key, receiver gets original data.
4. The receiver sends next data depending on the received message.

On the other hand, Dolev-Yao model [17] is assumed about behavior of an intruder as follows.

- Eavesdrop:

  Intruder can get sent data in a protocol.

- Falsify:

  Intruder can forcibly receive sent data to another participant, and change the data, and send the changed data to any participant.

- Decryption:

  If an intruder has corresponding decryption key, the intruder can get the original data.

- Using old data:

  An intruder knows old data that is sent and received in previous session, and use the old data for falsifying sent data. For example, in Needham-Schroeder Shared Key Protocol [24], an intruder can attack the protocol if the intruder has an old session key [13].

Targets of formalization are as follows.

- **Behavior of Participants**:

  It is a set of rules among actions in each step that represents as ‘if a participant receives a certain data, then the participant sends an other data.’

- **Behavior of an Intruder**:

  It represents a set of rules what actions an intruder performs in a communication process of a protocol. In the proposed method, behavior of an intruder is based on Dolev-Yao model [17].

- **Common Behavior among Participants and an Intruder**:

  It represents a set of rules except for behavior about sending and receiving data by participants and an intruder. For example, decryption of encrypted data is included in this target.

In this method, analysts formalize the above behaviors based on first order predicate strong relevant logics [6], [7]. In following explanation, individual variables p and p_i ($i \in \mathbb{N}$) represent participants or an intruder. Individual variables x and x_i ($i \in \mathbb{N}$) represent sent or received data. Individual variable k represents an encryption key. In order to be applied to various cryptographic protocols, I defined following predicates, functions, and individual constants that represent participants’ behavior or data in some cryptographic protocols. I defined following 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.

I defined following 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.

In addition, there are uniquely defined functions that are assigned to each participant's data.

I defined following individual constants.

- a, b: Persons

- i: An intruder.

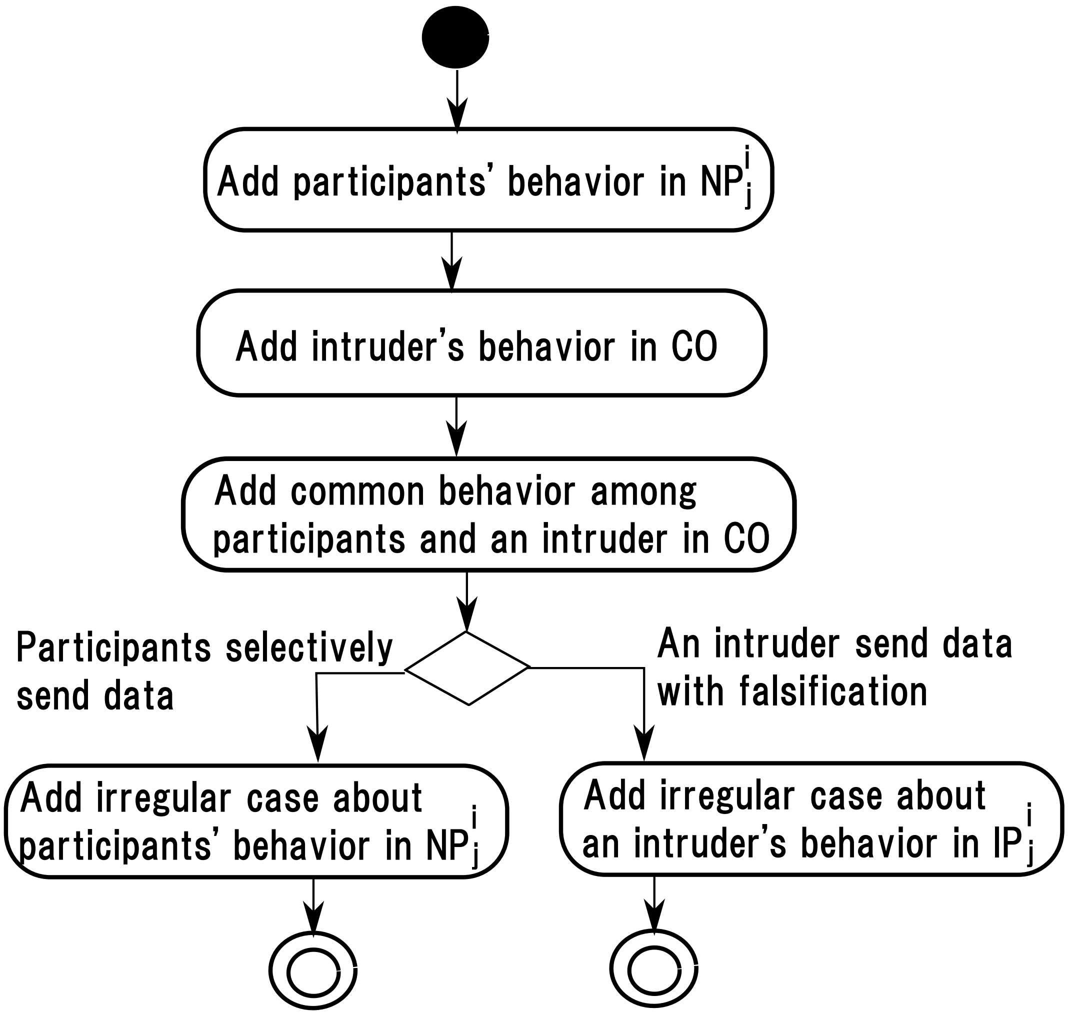

Add participants' behavior in NP_j^i
Add intruder's behavior in CO
Add common behavior among participants and an intruder in CO
Participants selectively send data
An intruder send data with falsification
Add irregular case about participants' behavior in NP_j^i
Add irregular case about an intruder's behavior in IP_j^i

Figure 3.1: Overview of tasks for formalization.

- *s*: A trusted server

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

In the proposed method, I assumed following premises about participants' and an intruder's behavior. At first, if received data that participants have is falsified, those participants can recognize falsification. In the proposed method, behavior of participants represents 'if a participant receives a certain data, then the participant sends an other data that is defined in the protocol.' Therefore, in following steps of formalization, it is assumed that participants verify whether received data, for example the signature and nonce, and so on, is falsified or not. 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 next data.

Figure 3.1 shows overview of tasks for formalization. In the proposed tasks of formalization, three sets of formulas are generated, Normal Path (NP), Irregular Path (IP), and Common Input (CO). NP_n^m is a set of logical formulas that represents that a participant correctly sends data in step n. Those logical formulas are generated by tasks in section 3.2.2. n represents the number of step. m represents the number of branches in case that participants send data selectively to another in step n. IP_n^m is a set of logical formulas that represents that an intruder sends data in step n. Those logical formulas are generated in section 4.2.5. m represents the number of branch by falsification. Finally, CO is a set of logical formulas that represents implicit participants' and an intruder's behavior. Those logical formulas are generated by tasks in section 3.2.3 and section 3.2.4.

3.2.2 Behavior of Participants

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

   (a) If step 1 of the protocol, use formulas (3.2), which means "if session starts with p_1 and p_2, p_1 sends data $x_1, \ldots, x_n$ to p_3".

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

   (b) If step 2 and beyond of the protocol, use the following formula (3.3), which means "if a participant p_1 receives data, p_1 sends next data to p_2."

   $$\begin{aligned} Recv(p_1, data(x_1, \ldots, x_m)) \Rightarrow (Parti(p_1) \Rightarrow \\ Send(p_1, p_2, data(x_1, \ldots, x_n))) \end{aligned} \tag{3.3}$$

2. Replace individual variables p_1 and p_2 of formulas in the previous task.

   (a) About formulas (3.2),

      i. If sender of corresponding step is a trusted server S, individual variable p_1 is replaced with s.

      ii. If receiver of corresponding step is a trusted server S, individual variable p_2 is replaced with s.

      iii. If sender of corresponding step is A, individual variable p_1 is not replaced.

      iv. If sender of corresponding step is B, individual variable p_1 is replaced with p_2.

      v. If receiver of corresponding step is A, individual variable p_2 is replaced with p_1.

vi. If receiver of corresponding step is B, individual variable p_2 is not replaced.

(b) About formulas (3.3),

i. If sender of corresponding step is a trusted server S, individual variable p_1 is replaced with s.

ii. If receiver of corresponding step is a trusted server S, individual variable p_2 is replaced with s.

iii. If sender of corresponding step is A, individual variable p_1 is not replaced.

iv. If sender of corresponding step is B, individual variable p_1 is replaced with p_2.

v. If receiver of corresponding step is A, individual variable p_2 is replaced with p_1.

vi. If receiver of corresponding step is B, individual variable p_2 is not replaced.

3. Replace individual variables $x_1, \cdots, x_n$ of formulas (3.2) and (3.3) with terms according to following rules corresponding step of the specification.

(a) If sent data Y_i is not encrypted, substitute a function or an individual variable depending on data types for x_i.

i. If Y_i is nonce of A, individual variable x_i is replaced with $nonce(p_1)$.

ii. If Y_i is nonce of B, individual variable x_i is replaced with $nonce(p_2)$.

iii. If Y_i is nonce of S, individual variable x_i is replaced with $nonce(s)$.

iv. If Y_i is identifier of A, individual variable x_i is replaced with $id(p_1)$.

v. If Y_i is identifier of B, individual variable x_i is replaced with $id(p_2)$.

vi. If Y_i is identifier of S, individual variable x_i is replaced with $id(s)$.

vii. If Y_i is timestamp of A, individual variable x_i is replaced with $tstamp(p_1)$.

viii. If Y_i is timestamp of B, individual variable x_i is replaced with $tstamp(p_2)$.

ix. If Y_i is timestamp of S, individual variable x_i is replaced with $tstamp(s)$.

x. If Y_i is session key SK, individual variable x_i is replaced with another individual variable k.

xi. If Y_i is others data, individual variable x_i is not replaced.

(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,

i. Substitute $enc(k, x'_1, \cdots, x'_n)$

ii. Replace k depending on key types.

A. If K is symmetric key of A and trusted server S, k is replaced with $symk(p_1, s)$.

B. If K is symmetric key of B and trusted server S, k is replaced with $symk(p_2, s)$.

C. If K is symmetric key of A and B, k is replaced with $symk(p_1, p_2)$.

D. If K is public key of A, k is replaced with $pk(p_1)$.

E. If K is public key of B, k is replaced with $pk(p_2)$.

F. If K is public key of a trusted server S, k is replaced with $pk(s)$.

G. If K is session key, k is not replaced.

4. Replace x'_i using funcion and individual variables as well as task in step (3).
5. In part of formulas $A_1 \Rightarrow A_2$ (A_1 and A_2 are formulas), if a individual variable is included only in A_1 or A_2, define an individual constant and replace the variable into the constant.
6. Add quantifier $\forall$ corresponded to individual variables k, x_i, and p_i in those formulas.
7. Add generated logical formulas that represent behavior of step i in NP^1_i.
8. 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 Behavior of an Intruder

Behavior of an intruder in the proposed method is based on Dolev-Yao model [17]. In this model, it is assumed that an intruder knows old data that is sent and received in the previous communication process, and eavesdropping, falsifying, decryption by a key that the intruder has. Formalization tasks of the rules are as follows.

1. Enumerate all functions $data(x_1, \ldots, x_n)$ in predicate $Send$ that are generated based on formulas (3.2) and (3.3) in section 4.2.2.)

2. 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 $data(x_1, \ldots, x_n)$.

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

4. Add generated formulas and a following formula that represents "if a participant sends x, an intruder gets it (by eavesdropping)"in CO.

   $\forall p_1 \forall p_2 \forall x (Send(p_1, p_2, x) \Rightarrow Get(i, x))$

3.2.4 Common Behavior among Participants and an Intruder

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

- If p_1 gets data $x_1, \cdots, x_n$ encrypted by p_1's symmetric key, p_1 gets original data (If a protocol that uses symmetric key).

  $\forall p_1 \forall p_2 \forall x_1 \ldots \forall x_n (Get(p_1, enc(symk(p_1, p_2), x_1,$

  $\cdots, x_n)) \Rightarrow Get(p_1, data(x_1, \cdots, x_n)))$

- If p gets data encrypted by p's public key, p gets original data (If a protocol that uses public key).

  $\forall p \forall x_1 \ldots \forall x_n (Get(p, enc(pk(p), x_1, \cdots, x_n)) \Rightarrow$

  $Get(p, data(x_1, \cdots, x_n)))$

- If p receives x, p gets it.

  $\forall p \forall x (Recv(p, x) \Rightarrow Get(p, x))$

- $symk(p_1, p_2)$ and $symk(p_2, p_1)$ are equal. (This formula needs to perform forward reasoning about symmetric key in the method.)

  $\forall p_1 \forall p_2 (Eq(symk(p_1, p_2), symk(p_2, p_1)))$

- A appears as a participant in target specification.

  $Parti(\alpha)$ where α is a individual constant that represents a person or a trusted server.

- An intruder is not a participant.

  $\neg Parti(i)$

3.2.5 Irregular Case

Analysts generate logical formulas that are corresponded to each case of following participants' and an intruder's actions.

C1: Case that a participant receives sent data correctly.

C2: Case that an intruder receives sent data forcibly and send falsified data to any participant.

If a participant receives data correctly, generated logical formulas are added in $NP_j^1, \dots, NP_j^n$ where n is the number of data that is selectively sent by participants in step j. If an intruder receives sent data forcibly and send falsified data to any participant, generated logical formulas are added in $IP_j^1, \dots, IP_j^n$ as well where n is the number of falsified data can be sent by the intruder in step j.

In following explanation, p, p_1, and p_2 are individual constants of participants that perform sending or receiving, and $x, x_1, \dots,$ and x_n ($n \in \mathbb{N}$) are individual constants of data that are sent in step j.

Analysts perform following tasks in case that such branches are occurred in step j.

1. If $Send(i, p, x)$ is deduced, analysts add $Recv(p, x)$ in NP_j^i.
2. If $Send(p_1, p_2, x)$ is deduced, analysts add following formulas respectively.
   - Analysts add $Recv(p_2, x)$ in NP_j^1.
   - Analysts add logical formulas that are generated in task 4.
3. If $Send(p, i, x)$ is deduced, analysts add logical formulas that are generated in task 4.
4. Analysts generate logical formulas that represent an intruder sends data with falsification by following tasks.
   (a) Enumerate participants that receive any data after step j (Y in specification 3.1). If there is no participant, tasks of forward reasoning is over.
   (b) Repeat following tasks in each enumerated participant.
      i. Extract one participant as receiver p', and check the number of step (after $X \rightarrow Y$ in specification 3.1).
      ii. Generate logical formulas that represent sent data in the step based on task 2 and 3 of section 4.2.2 (with replaced individual variable p_i into individual constant).
      iii. Replace each data into data that an intruder owns.
   (c) Generate logical formulas $Send(i, p', data_1), \dots, Send(i, p', data_n)$, and add those formulas with $Recv(i, x)$ in $IP_j^1, \dots, IP_j^n$, respectively ($data_1, \cdots$, and $data_n$ are the result of falsification, and n is number of data that the an intruder can send.).

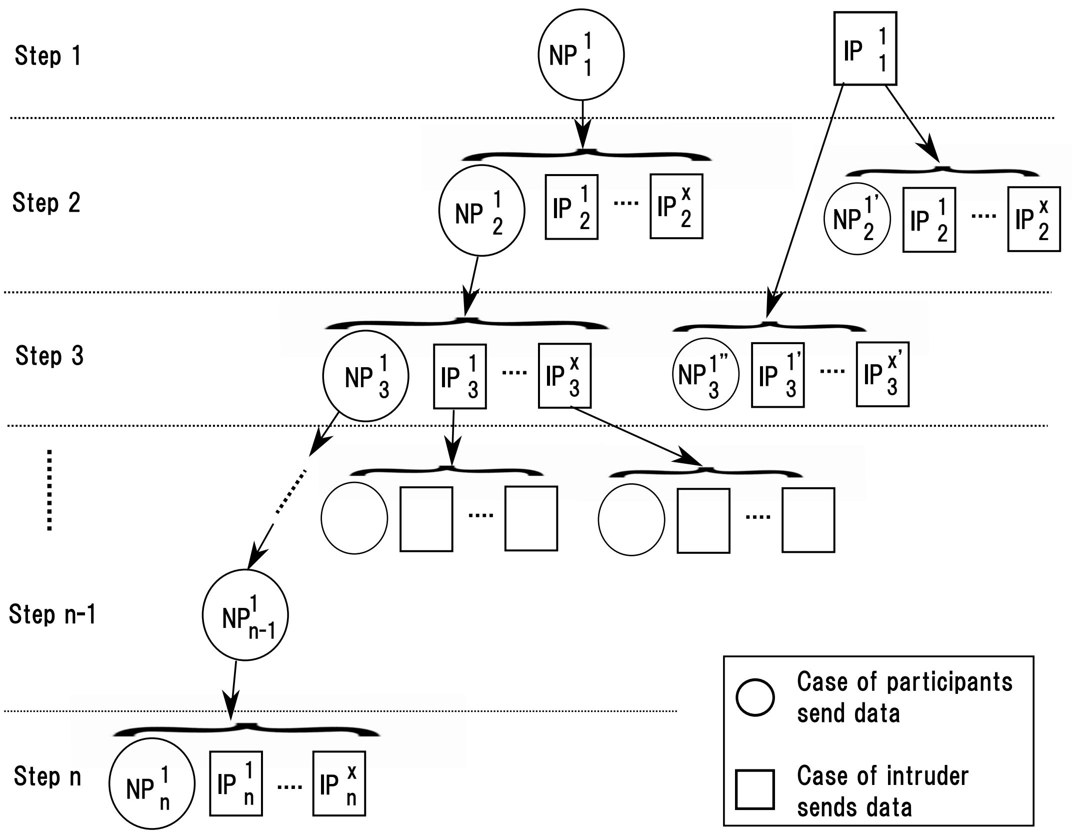


Figure 3.2: Branches by cases that participants and an intruder sends data.

Figure 3.2 shows overview of branches by cases that participants and an intruder sends data. In this paper, I call a route of forward reasoning 'path.' In above conditions of falsification, complexity of the tasks is increased by the number of logical formulas IP_n^m. If there is a case that a participant sends data selectively to another, the number of logical formulas NP_n^m also increase the complexity.

3.3 Forward Reasoning

In the proposed method, analysts use FreeEnCal [9] in order to perform forward reasoning automatically. Analysts use sets of generated logical formulas NP_i^j or $IP_i^{j'}$, and CO that are generated in section 3.2, 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.

In tasks of forward reasoning, analysts use FreeEnCal for forward reasoning about each step of the target protocol. Figure 3.3 shows overview of tasks of forward reasoning. As the number of step is i, and the number of data that participants or an intruder can send is n, analysts divide into cases that NP_i^j or $IP_i^{j'}$ ($1 \le j \le n$, $1 \le j' \le n'$), is used as input of FreeEnCal.

In the proposed method, an intruder does not forge sending data if participants can

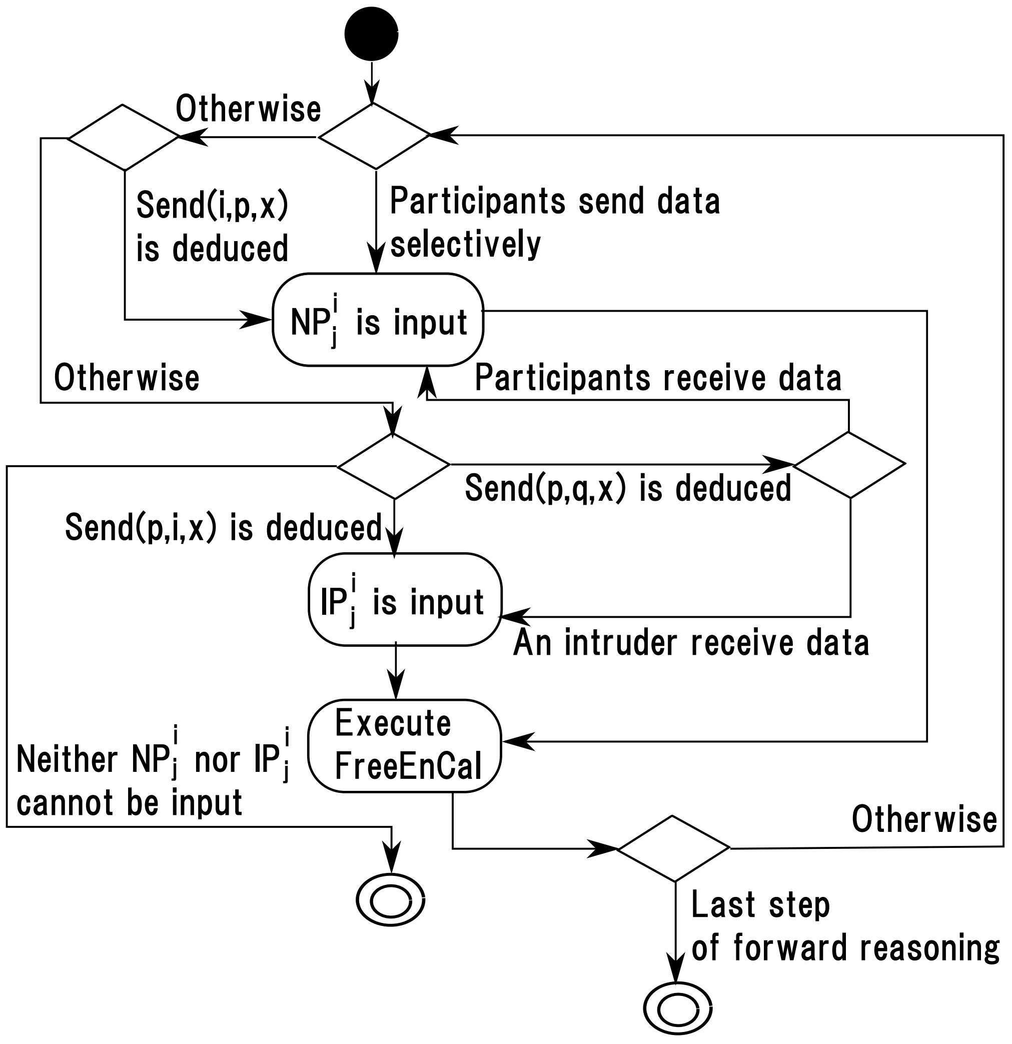


Figure 3.3: Overview of tasks of forward reasoning.

recognize falsification, and participants know what kinds of data are derived to them as next data. 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 in section 3.4.

3.4 Tasks of Analysis

Analysts check whether formulas that represent success of attack are included in each DF that is the case of completion of tasks of forward reasoning.

1. Check whether those formulas that represent success of attack are included in

deduced logical formulas after the removal.

- Formulas that represent successful completion of the protocol (in forward reasoning about last step of target protocol, an atomic formula that starts "*Recv*" is deduced).
- Formulas that represent obtaining session key as secret data by an intruder "$Get(i, sesk)$".
- Formulas that represent that an intruder received sent data to a participant "$Send(p_1, p_2, anydata)$," and a participant received falsified data by an intruder $Recv(i, anydata)$, and $Send(i, p_2, anydata')$ ($anydata'$ is falsified data from $anydata$).

2. If following both of formulas are included, it can be said that attacks by an intruder succeed in the target cryptographic protocol.

   - Formulas that represent successful completion of the protocol.
   - Formulas that represent that an intruder gets session key, or any participant received falsified data by an intruder.

In figure 3.4, formulas that represent successful completion of a protocols mean that any participant sent or got data in forward reasoning about the last step of protocol. Analysts check whether an atomic formula that starts "*Recv*" is deduced or not in forward reasoning about last step of target protocol. Eavesdropping is represented as $Get(i, x)$ (x is secret data). In key exchange protocols, a session key is secret data. Falsifying means that sent data to a participant is received by an intruder, and the intruder sends another data to the participant. It is represented as $Send(p, q, x)$, $Receive(i, x)$, and $Send(i, q, x')$ in deduced logical formulas.

3.5 Applying the Proposed Method to Otway-Rees Protocol

I show that the proposed method can be applied to key exchange protocols. In order to show it, I chose Otway-Rees protocol [27] as a case of formal analysis of key exchange protocols by the proposed method. Specification of this protocol is as follows. In the specification, M is data that is uniquely defined in the protocol.

1. $A \rightarrow B : M, A, B, \{N_A, M, A, B\}_{K_{AS}}$
2. $B \rightarrow S : M, A, B, \{N_A, M, A, B\}_{K_{AS}}, \{N_B, M, A, B\}_{K_{BS}}$
3. $S \rightarrow B : M, \{N_A, SK\}_{K_{AS}}, \{N_B, SK\}_{K_{BS}}$
4. $B \rightarrow A : M, \{N_A, SK\}_{K_{AS}}$

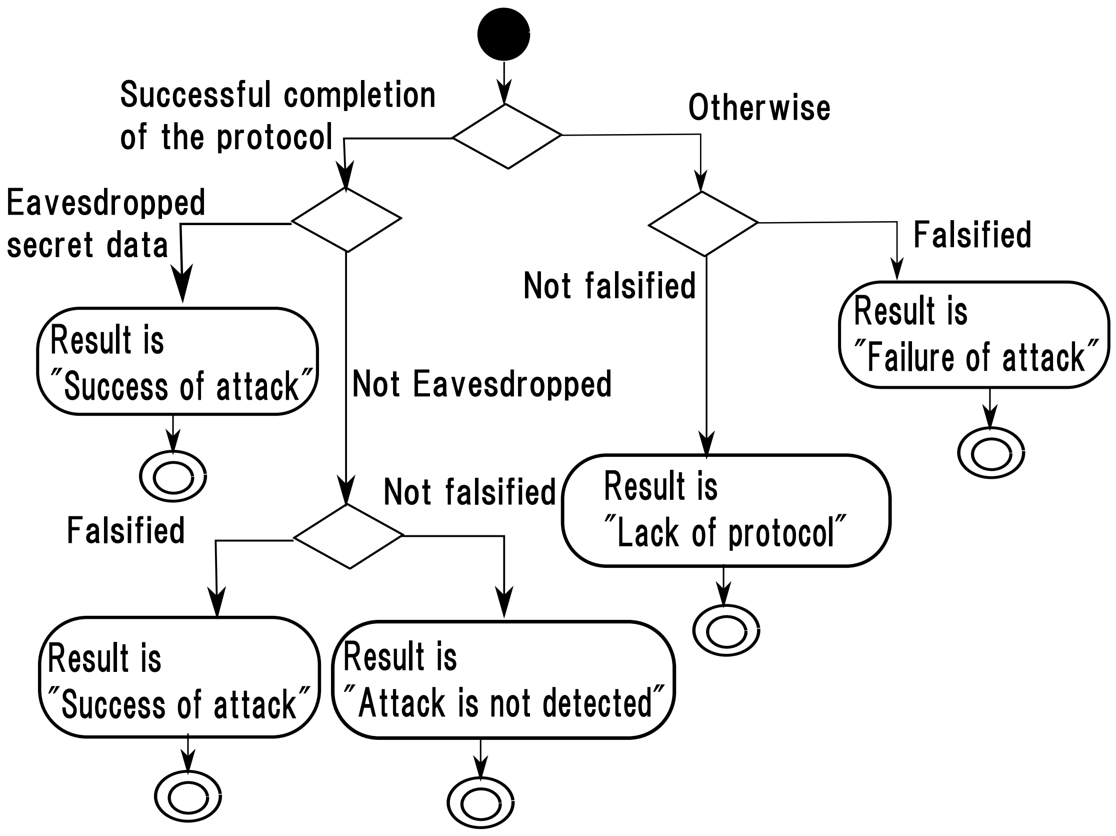

Successful completion of the protocol
Otherwise
Eavesdropped secret data
Falsified
Not falsified
Result is "Success of attack"
Not Eavesdropped
Result is "Failure of attack"
Not falsified
Result is "Lack of protocol"
Falsified
Result is "Success of attack"
Result is "Attack is not detected"

Figure 3.4: Overview of tasks of analysis.

In Otway-Rees protocol, two flaws have been pointed out. At first, an intruder can get the session key by falsifying encrypted data of a participant into that of the intruder [5]. Concretely, attack succeeds by following steps for sending and receiving data.

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

2'. $B \rightarrow I(S) : M, A, B, \{N_A, M, A, B\}_{K_{AS}}, \{N_B, M, A, B\}_{K_{BS}}$

2". $I(B) \rightarrow S : M, A, I, \{N_A, M, A, B\}_{K_{AS}}, \{N_I, M, A, B\}_{K_{IS}}$

3'. $S \rightarrow I(B) : M, \{N_A, SK\}_{K_{AS}}, \{N_I, SK\}_{K_{IS}}$

4'. $I(B) \rightarrow A : M, \{N_A, SK\}_{K_{AS}}$

Second, if an intruder resend encrypted data of A, A mis-recognizes the received data M, A, B as the session key [10]. Concretely, attack succeeds by following steps for sending and receiving data.

1. $A \rightarrow I(B) : M, A, B, \{N_A, M, A, B\}_{K_{AS}}$

4'. $I(B) \rightarrow A : M, \{N_A, M, A, B\}_{K_{AS}}$

As the result of formal analysis by the proposed method, 32 paths were generated, and 73,131 logical formulas were deduced. I could re-detected fatal actions of attacks that represent success of attack for those two flaws. About first flaw, $Get(i, sesk)$ and $Get(a, data(nonce(a), sesk))$ were included in the deduced logical formulas. $Get(i, sesk)$ means that an intruder gets a session key that must not be known to the intruder.
$Get(a, data(nonce(a), sesk))$ means that A gets nonce of A and a session key that A should get in the last step of protocol. Therefore, these logical formulas mean that A completed all steps of protocol without recognizing that the intruder gets the session key. It is equal to first flaw that is pointed out. About second flaw, deduced logical formulas included $Get(a, data(nonce(a), m, id(a), id(b)))$. This formula means that A completed all steps of protocol. Originally, $Get(a, data(nonce(a), sesk))$ should be deduced that represents that A completed all steps of protocol. A knows nonce of A previously, but does not know the session key. Therefore, if A receives M, A, B, it is possible that A does not recognize falsification of data by an intruder. Furthermore, it is possible that A mis-recognizes the received data M, A, B as session key. It is equal to second flaw that is pointed out.

In case of Otway-Rees protocol, I could re-detect fatal actions of attacks that represent success of attack for two flaws of this protocol that are pointed out. Therefore, I showed that the proposed method can detect fatal actions of attacks that represent success of attack.

In key exchange protocols, it is common that any participant sends any data in each step as specification in 3.1. Sender and sent data in each step are different for key exchange protocols. Therefore, it can be said that the proposed method can be applied to other key exchange protocols.

Chapter 4

Extending Proposed Formal Analysis Method with Reasoning for Key Exchange Protocols

4.1 Comparison of Cryptographic Protocols

Current formal analysis method with reasoning considers only key exchange protocols [33], but not other cryptographic protocols. At first, I compared cryptographic protocols that have been proposed [26], [31], based on participants' behaviors and the number of participants. As the result, I found five differences among cryptographic protocols. Table 4.1 shows the differences of participants' behaviors and the number of participants. In table 4.1, following differences from D1 to D5 are represented in cryptographic protocols.

D1: There is a case that number of participants is 3 or more.

In most cryptographic protocols, number of participants is 2. However, there are some cryptographic protocols that 3 or more participants can participate, for example, mental poker and secure multiparty computation, and so on.

D2: There is 1 or more trusted server.

In some cryptographic protocols, there is 1 or more trusted server in order to success those protocols correctly, for example, key exchange, authentication, and so on.

D3: Any participant sends data selectively to another.

In most cryptographic protocols, sent data by participants is uniquely determined. However, in some cryptographic protocols, a participant sends data selectively to another participant, for example, coin flips, zero knowledge proofs, and so on.

D4: Any participant sends data to multiple participants.

In most cryptographic protocols, a participant sends data to one another participant. However, in some cryptographic protocols, a participant sends data to multiple participants at the same time, for example, secret splitting, bit commitment, and so on.

D5: There is a case that participants get another data by calculation for collected data in a protocol.

In some cryptographic protocols, there is a case that if any participant receives some data, the participant gets new data. For example, in secret splitting protocols if a participant gets all split data, the participant can get secret data.

According to table 4.1, key exchange protocols have only feature **D1**, **D2**, and **D5**. Thus, the proposed method does not consider **D3** and **D4**. I should extend the method to deal with **D3** and **D4**.

Kind of Protocol	**D1**	**D2**	**D3**	**D4**	**D5**
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 4.1: Comparison of behaviors and the number of participants in cryptographic protocols

4.2 Formalization of Cryptographic Protocols

4.2.1 Overview of Formalization

As well as the proposed method in section 3.2, analysts perform formalization from specification that is represented as eq. 3.1. In cryptographic protocols, it is common that any participant sends and receives data in each step. Furthermore, defined data in section 3.1 is used in some cryptographic protocols. Therefore, defined predicates, functions, individual variables, and individual constants in section 3.1 are used in the extended method.

Following tasks are new tasks in our extended formalization method.

1. In order to corresponding to **D1**, I extended tasks 1 and 2 in section 4.2.2 by adding the case that there are 3 or more participants. I added definition of individual constants $a, b, \dots, h, a_1, \dots, a_n$ ($n \in \mathbb{N}$) that represent persons （If the number of person is over 8, those individual constants are represented as $a_1, \dots, a_n$ in order to distinguish persons from an intruder).

2. In order to corresponding to **D2**, I extended tasks 2 in section 4.2.2 by adding the case that there are multiple trusted servers. I added definition of individual constants $s_1, \dots, s_n$ ($n \in \mathbb{N}$) that represent trusted servers.

3. In order to corresponding to **D3**, I extended task 1 in section 4.2.5 by adding the case that a participant selectively sends data.

4. In order to corresponding to **D4**, I extended task 1 in section 4.2.2 by adding the case that a participant sends data to multiple participants

5. In order to corresponding to **D5**, I added task of generating logical formulas that participants get new data by calculating got multiple data in section 4.2.4.

4.2.2 Behavior of Participants

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

   (a) If step 1 of the protocol, or while all previous senders of data are same in step 1, use the following formula (4.1), which means "if a communication process starts with p_1 and p_2, p_1 sends data $x_1, \dots, x_n$ to p_2". If a participant selectively sends data in corresponding step, logical formulas about corresponding step are generated in section 4.2.5.

   $$Start(p_1, p_2) \Rightarrow Send(p_1, p_2, data(x_1, \dots, x_n)) \tag{4.1}$$

   (b) If step 2 and beyond of the protocol, and sender of data is not same in step 1, use the following formula (4.2), which means "if a participant p_1 receives data, p_1 sends next data to p_2." But p_i and x_i are individual variables, and n and m are the number of sent or received data. If a participant selectively sends data in corresponding step, logical formulas about corresponding step are generated in section 4.2.5.

$$Recv(p_1, data(x_1, \ldots, x_m)) \Rightarrow (Parti(p_1) \Rightarrow Send(p_1, p_2, data(x_1, \ldots, x_n))) \tag{4.2}$$

2. Replace individual variables p_1 and p_2 of formulas in the previous task.
   (a) About formulas (4.1),
      i. 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.
      ii. If receiver of corresponding step is a trusted server $S_1, S_2, \ldots,$ or S_n, individual variable p_2 is replaced with $s_1, s_2, \ldots,$ or s_n, respectively.
      iii. If sender of corresponding step is the i^{th} participant, individual variable p_1 is replaced with p_i.
      iv. If receiver of corresponding step is the i^{th} participant, individual variable p_2 is replaced with p_i.
   (b) About formulas (4.2),
      i. 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.
      ii. If receiver of corresponding step is a trusted server $S_1, S_2, \ldots,$ or S_n, individual variable p_2 is replaced with $s_1, s_2, \ldots,$ or s_n, respectively.
      iii. If sender of corresponding step is the i^{th} participant, individual variable p_1 is replaced with p_i.
      iv. If receiver of corresponding step is the i^{th} participant, individual variable p_2 is replaced with p_i.
3. Replace individual variables $x_1, \cdots, x_n$ of formulas (4.1) and (4.2) with terms according to following rules corresponding step of the specification.
   (a) If sent data Y_i is not encrypted, substitute a function or an individual variable depending on data types for x_i.
      i. If Y_i is data that the i^{th} participant owns, use function (for example f). After that, individual variable x_i is replaced with $f(p_i)$.
      ii. If Y_i is data that a trusted server $S_1, S_2, \ldots,$ or S_n owns, use function (for example f). After that, individual variable x_i is replaced with $f(s_1), f(s_2), \ldots,$ or $f(s_n)$, respectively.

iii. If Y_i is data that no one owns, x_i is replaced with 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,

i. Substitute $enc(k, x_1', \cdots, x_n')$

ii. Replace k depending on key types.

A. If K is symmetric key of the i^{th} participant and a trusted server $S_1, S_2, \ldots,$ or S_n, k is replaced with $symk(p_i, s_1), symk(p_i, s_2), \ldots,$ or $symk(p_i, s_n)$, respectively.

B. If K is symmetric key of the i^{th} and the j^{th} participant, k is replaced with $symk(p_i, p_j)$.

C. If K is public key of the i^{th} participant, k is replaced with $pk(p_i)$.

D. If K is public key of a trusted server $S_1, S_2, \ldots,$ or S_n, k is replaced with $pk(s_1), pk(s_2), \ldots,$ or $pk(s_n)$, respectively.

E. If K is other kinds of key, k is not replaced.

4. In part of formulas $A_1 \Rightarrow A_2$ (A_1 and A_2 are 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 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.

4.2.3 Behavior of an Intruder

Behavior of an intruder in the proposed method is based on Dolev-Yao model [17]. In this model, it is assumed that an intruder knows old data that is sent and received in the previous communication process, and eavesdropping, falsifying, decryption by a key that the intruder has. Formalization tasks of the rules are as follows.

1. Enumerate all functions $data(x_1, \ldots, x_n)$ in predicate $Send$ that are generated based on formulas (4.1) and (4.2) in section 4.2.2.)

2. 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 $data(x_1, \ldots, x_n)$.

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

4. Add generated formulas and a following formula that represents "if a participant sends x, an intruder gets it (by eavesdropping)" in CO.

   $\forall p_1 \forall p_2 \forall x (Send(p_1, p_2, x) \Rightarrow Get(i, x))$

4.2.4 Common Behavior among Participants and an Intruder

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

- If p_1 gets data $x_1, \cdots, x_n$ encrypted by p_1's symmetric key, p_1 gets original data (If a protocol that uses symmetric key).

  $\forall p_1 \forall p_2 \forall x_1 \ldots \forall x_n (Get(p_1, enc(symk(p_1, p_2), x_1,$

  $\cdots, x_n)) \Rightarrow Get(p_1, data(x_1, \cdots, x_n)))$

- If p gets data encrypted by p's public key, p gets original data (If a protocol that uses public key).

  $\forall p \forall x_1 \ldots \forall x_n (Get(p, enc(pk(p), x_1, \cdots, x_n)) \Rightarrow$

  $Get(p, data(x_1, \cdots, x_n)))$

- If p receives x, p gets it.

  $\forall p \forall x (Recv(p, x) \Rightarrow Get(p, x))$

- $symk(p_1, p_2)$ and $symk(p_2, p_1)$ are equal. (This formula needs to perform forward reasoning about symmetric key in the method.)

  $\forall p_1 \forall p_2 (Eq(symk(p_1, p_2), symk(p_2, p_1)))$

- A appears as a participant in target specification.

  $Parti(\alpha)$ where α is a individual constant that represents a person or a trusted server.

- An intruder is not a participant.

  $\neg Parti(i)$

Furthermore, in order to represent a participant previously gets data, analysts add following formulas with replacing x into an individual constant that represents a participant previously gets data in target cryptographic protocol.

- $Get(s_i, x)$
- $\forall p_1 \forall p_2 (Start(p_1, p_2) \Rightarrow Get(p_1, x))$ (If first sender previously gets data x)
- $\forall p_1 \forall p_2 (Start(p_1, p_2) \Rightarrow Get(p_2, x))$ (If first receiver previously gets data x)

In order to represent that p gets another data by calculation for collected data in a protocol (for example $getdata_1, \ldots, getdata_n$), analysts add following formula with replacing $getdata_1, \ldots, getdata_n$, and $getdata'$ into an individual constant that represents data that a participant should collect ($getdata_1, \ldots, getdata_n$) in order to get another data by calculation ($getdata'$). (For example, in secret splitting protocol, if a participant gets all split data, the participant gets secret data.)

$\forall p((Get(p, getdata_1) \wedge \cdots \wedge Get(p, getdata_n) \Rightarrow Get(p, getdata')))$
where $getdata_i$ and $getdata'$ are individual constants that represent data.

4.2.5 Irregular Case

Analysts generate logical formulas that are corresponding to each case of participants' and an intruder's actions. If participants selectively send data, generated logical formulas are added in $NP_j^1, \ldots, NP_j^n$ where n is the number of data that is selectively sent by participants in step j. If an intruder sends data with falsification, generated logical formulas are added in $IP_j^1, \ldots, IP_j^n$ as well where n is the number of falsified data can be sent by the intruder in step j.

In following explanation, p, p_1, and p_2 are individual constants of participants that perform sending or receiving, and $x, x_1, \ldots,$ and x_n ($n \in \mathbb{N}$) are individual constants of data that are sent in step j. In following tasks, it is assumed that case of falsification in each kind of data is unique because participants check only whether received data is same to that of participants'. For example, it is possible that an intruder has multiple nonces, but the result of falsification is same in case of which nonce is used to falsification, because participants check only whether received nonce is same to that of participants'.

Analysts perform following tasks in case that such branches are occurred in step j.

1. If a participant sends data selectively to an other participant in forward reasoning about step j, analysts add $Send(p_1, p_2, x_1), \ldots, Send(p_1, p_2, x_n)$ in $NP_j^1, \ldots, NP_j^n$, respectively.
2. If $Send(i, p, x)$ is deduced, analysts add $Recv(p, x)$ in NP_j^i.
3. If $Send(p_1, p_2, x)$ is deduced, analysts add following formulas respectively.
    - Analysts add $Recv(p_2, x)$ in NP_j^1.
    - Analysts add logical formulas that are generated in task 5.
4. If $Send(p, i, x)$ is deduced, analysts add logical formulas that are generated in task 5.

5. Analysts generate logical formulas that represent an intruder sends data with falsification by following tasks.
    (a) Enumerate participants that receive any data after step j (Y in specification 3.1). If there is no participant, tasks of forward reasoning is over.
    (b) Repeat following tasks in each enumerated participant.
        i. Extract one participant as receiver p', and check the number of step (after $X \rightarrow Y$ in specification 3.1).
        ii. Generate logical formulas that represent sent data in the step based on task 2 and 3 of section 4.2.2 (with replaced individual variable p_i into individual constant).
        iii. Replace each data based on following conditions of falsification. If any data cannot be replaced, continue to next participant. (In following conditions, for example, if an intruder sends N_B, A (indicator) to A, N_B can be falsified because A does not have N_B, and indicator that A owns cannot be falsified.)
            - Identifier: If receiver p' owns the identifier, it cannot be falsified. Otherwise, it can be falsified into an intruder's identifier.
            - Timestamp: If receiver p' owns the timestamp, it cannot be falsified. Otherwise, it can be falsified into an intruder's timestamp.
            - Nonce: If receiver p' owns the nonce, it cannot be falsified. Otherwise, it can be falsified into an intruder's nonce.
            - Incremented data: If receiver p' owns the data that is not incremented, it cannot be falsified. Otherwise, it can be falsified based on condition for kind of the data.
            - Other data that is uniquely defined in a protocol: If anyone owns the data, it can be falsified into an intruder's data without changing kind of data. If no one owns the data, it can be falsified into old data that an intruder has.
            - Encrypted data: If the data is encrypted by key of receiver p', it can be falsified into data with satisfying the condition that is encrypted by key of p'. If the data is encrypted by the other's key, it can be also falsified into data with satisfying the condition that is encrypted by an intruder's key.
            - Data with signature: The contents of the data with signature can be falsified by the condition for each kind of the data, without changing the signature.
    (c) Generate logical formulas $Send(i, p', data_1), \ldots, Send(i, p', data_n)$, and add those formulas with $Recv(i, x)$ in $IP_j^1, \ldots, IP_j^n$, respectively ($data_1, \cdots$, and $data_n$ are the result of falsification, and n is number of data that the an intruder can send.).

Figure 3.2 shows overview of branches by cases that participants and an intruder sends data. In above conditions of falsification, complexity of the tasks is increased by the number of logical formulas IP_n^m. If there is a case that a participant sends data selectively to another, the number of logical formulas NP_n^m also increase the complexity. However, the task of forward reasoning can be finished by finite time because case of falsification in each kind of data is unique.

4.3 Forward Reasoning

In the proposed method, analysts use FreeEnCal [9] in order to perform forward reasoning automatically. Analysts use sets of generated logical formulas NP_i^j or $IP_i^{j'}$, and CO that are generated in section 4.2, 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.

In tasks of forward reasoning, analysts use FreeEnCal for forward reasoning about each step of the target protocol. As the number of step is i, and the number of data that participants or an intruder can send is n, analysts divide into cases that NP_i^j or $IP_i^{j'}$ ($1 \leq j \leq n$, $1 \leq j' \leq n'$), is used as input of FreeEnCal.

In the proposed method, an intruder does not forge sending data if participants can recognize falsification, and participants know what kinds of data are derived to them as next data. 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 in section 4.4.

4.4 Tasks of Analysis

Analysts check whether formulas that represent success of attack are included in each DF that is the case of completion of tasks of forward reasoning.

1. Check whether the formulas that represent success of attack are included in deduced logical formulas after the removal.

    - Formulas that represent successful completion of the protocol (in forward reasoning about last step of target protocol, an atomic formula that starts "*Recv*" is deduced).
    - Formula that represent an intruder obtains secret data "$Get(i, secretdata)$".
    - Formulas that represent that an intruder received sent data to a participant "$Send(p_1, p_2, anydata)$," and a participant received falsified data by an intruder $Recv(i, anydata)$, and $Send(i, p_2, anydata')$ ($anydata'$ is falsified data from $anydata$).

2. If following both of formulas are included, it can be said that attacks by an intruder succeed in the target cryptographic protocol.

    - Formulas that represent successful completion of the protocol.

    - Formulas that represent that an intruder gets secret data, or any participant received falsified data by an intruder.

In figure 3.4, formulas that represent successful completion of a protocols mean that any participant sent or got data in forward reasoning about the last step of protocol. Analysts check whether an atomic formula that starts "*Recv*" is deduced or not in forward reasoning about last step of target protocol. Eavesdropping is represented as $Get(i, x)$ (x is secret data). Falsifying means that sent data to a participant is received by an intruder, and the intruder sends another data to the participant. It is represented as $Send(p, q, x)$, $Receive(i, x)$, and $Send(i, q, x')$ in deduced logical formulas.

Chapter 5

Demonstration of Effectiveness

5.1 Method of Demonstration

I show that the proposed method can be applied to various cryptographic protocols. According to table 4.1 in section 4, difference in each cryptographic protocol is what feature the cryptographic protocol has in those five differences. Therefore, if formal analysis of some cryptographic protocols that include those five differences can be performed, it can be said that the extended method can also be applied to other than key exchange protocols.

In order to show it, I chose secret splitting protocol [31] and coin flips protocol [31], as cases of formal analysis that includes those five differences by the proposed method.

5.2 Case in Secret Splitting Protocols

Secret splitting protocols [31] 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, secret information is split into several parts of data. After that, administrator of secret information sends each split data to different participants. Each participant can restore secret information by collecting each split data. In secret splitting protocol, all split data must not be got by an intruder because an intruder can restore secret information, and must not be falsified because participants cannot restore secret information. As table 4.1 in section 4, secret splitting protocols have feature **D1**, **D2**, **D4**, and **D5**.

As the first step for formal analysis of secret splitting protocols, I prepared three secret splitting protocols as the case of formal analysis. In those three specification of secret splitting protocols, $data1$ and $data2$ are split data to send participants of protocol. Specification of first protocol is below.

Specification of 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 possibilities, 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

Step 1 $A \rightarrow I(S) : A, B$

Step 2 $I(A) \rightarrow S : I, I$

Step 3 $S \rightarrow I : \{data1\}_{K_{IS}}$

Step 4 $S \rightarrow I : \{data2\}_{K_{IS}}$

I 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

Step 1 $S \rightarrow A : \{data1, B\}_{K_{AS}}$

Step 2 $S \rightarrow B : \{data2, A\}_{K_{BS}}$

The second protocol has a flaw. In behavior of an intruder based on Dolev-Yao model [17] (section 4), 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

Step 1 $S \rightarrow I(A) : \{data1, B\}_{K_{AS}}$

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

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

Step 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)

Step 1 $S \rightarrow I(A) : \{data1, B\}_{K_{AS}}$

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

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

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

Step 1 $S \rightarrow A : \{data1, B\}_{K_{AS}}$

Step 2 $S \rightarrow I(B) : \{data2, A\}_{K_{BS}}$

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

I improved the first protocol in order to prevent falsifying of an intruder that is represented as third protocol. Specification of third protocol is below.

Specification of third protocol

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

Step 2 $B \rightarrow S : A, N_A, B, N_B$

Step 3 $S \rightarrow A : \{data1, B, N_A\}_{K_{AS}}$

Step 4 $S \rightarrow B : \{data2, A, N_B\}_{K_{BS}}$

In order to show that the extended method in section 4 can be applied to cryptographic protocols, I analyzed those three secret splitting protocols based on the extended formal analysis method with reasoning. At first, I formalized specification of each protocol based on section 4. After that, I performed forward reasoning as the premises. Finally, I detected fatal actions of attacks that represent success of attacks for flaws of those protocols from deduced logical formulas by forward reasoning.

About first protocol, I performed formal analysis based on the proposed method. As the result, 20 paths were generated and total 5086 logical formulas were deduced. In the deduced logical formulas, 5 logical formulas $Recv(s, data(id(i), id(i)))$, $Send(s, i, enc(symk(i, s), data1))$, $Send(s, i, enc(symk(i, s), data2))$, $Get(i, data1)$ and $Get(i, data2)$ were included. $Get(i, data1)$ and $Get(i, data2)$ mean success of attack that an intruder could get each split data $data1, data2$. As the result, the intruder can get secret data. $Recv(s, data(id(i), id(i)))$, $Send(s, i, enc(symk(i, s), data1))$, and $Send(s, i, enc(symk(i, s), data2))$ mean that a trusted server S finished all tasks because the number of receiving and sending is 1 and 2 respectively in the specification. This is equal to the flaw of first protocol.

About second protocol, I performed formal analysis based on the proposed method As the result, 4 paths were generated and total 3250 logical formulas were deduced. In the deduced logical formulas, 4 logical formulas $Get(a, data(old(data1), id(b)))$, $Get(b, data(old(data2), id(a)))$, $Recv(a, enc(symk(a, s), old(data1), id(b)))$ and $Recv(b, enc(symk(b, s), old(data2), id(a)))$, were included in deduced logical formulas. $Recv(a, enc(symk(a, s), old(data1), id(b)))$ means that A finished all tasks and $Recv(b, enc(symk(b, s), old(data2), id(a)))$ means that B finished all tasks because the number of receiving is 1 in the specification. $Get(a, data(old(data1), id(b)))$ and $Get(b, data(old(data2), id(a)))$ mean that A and B get data that an intruder falsified without recognizing the attack because the received split data has no meaning. As well as the result, logical formulas that represent that only one data is falsified were also deduced, for example, $Get(a, data(data1, id(a)))$ and $Get(b, data(old(data2), id(a)))$, or $Get(a, data(old(data1), id(a)))$, $Get(b, data(data2, id(a)))$. This is equal to the flaw of second protocol.

About third protocol, I performed formal analysis based on the proposed method. As the result, 14 paths were generated and total 16431 logical formulas were deduced. Fatal actions of attack that represent success of attack for flaws were not detected in deduced logical formulas but I could perform tasks of the proposed method to the end.

In cryptographic protocols that have feature **D1**, **D2**, **D4**, and **D5**, behavior of participants is common in each protocol but only sent data in the common behavior is different. Therefore, the proposed method can be applied to cryptographic protocols that have feature **D1**, **D2**, **D4**, and **D5**.

5.3 Case in Coin Flips Protocols

Coin flips protocols [31] 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. Coin flips protocols have behavior of **D3** that is explained in section 4.

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

I performed formal analysis of the coin flips protocol based on the proposed method. As the result, 8 paths were generated and total 6206 logical formulas were deduced. Fatal actions of attacks that represent success of attack for flaws were not detected in deduced logical formulas but I could perform tasks of the proposed method to the end.

In cryptographic protocols that have feature **D3**, behavior of participants is common in each protocol but only sent data in the common behavior is different. Therefore, the proposed method can be applied to cryptographic protocols that have feature **D3**.

Chapter 6

Supporting Tools for Proposed Method

6.1 Requirement Analysis

The concrete steps for formal analysis method with reasoning is proposed in chapter 4. However, it is time-consuming task for analysts to manually perform formal analysis based on the proposed method. As the result, it is possible for analysts to make mistakes in tasks in the proposed method. Therefore, supporting tools are needed in order to easily perform the proposed method for analysts. I enumerated following 5 requirements for supporting tools for the proposed method.

R1: Supporting tools must support tasks that are time-consuming for analysts.

- **R1.1:** Supporting tools must support tasks for formalization of a target cryptographic protocol.
- **R1.2:** Supporting tools must support tasks for forward reasoning from the result of formalization as premises.
- **R1.3:** Supporting tools must support tasks for analysis of deduced logical formulas by forward reasoning.

It is time-consuming task for analysts to perform tasks of formalization, forward reasoning, and analysis in the proposed method. **R1** is a basic requirement as supporting tools for the proposed method.

R2: Supporting tools must systematically store and manage specification of cryptographic protocols for input.

- **R2.1:** Supporting tools must store and manage the specification for each users.
- **R2.2:** Supporting tools must store and manage the specification for each protocols.
- **R2.3:** Supporting tools must store and manage the specification for each creation times.

It is possible that analysts perform formal analysis of many cryptographic protocols. As the result, it is time-consuming task for analysts to search only a target cryptographic protocol in many of those. Therefore, supporting tools must satisfy **R2** to support formal analysis of many cryptographic protocols by analysts.

R3: Supporting tools must make possible for analysts to use specification of cryptographic protocols for input as necessary.

R3.1: Supporting tools must make possible for analysts to read specification of cryptographic protocols for input as necessary.

R3.2: Supporting tools must make possible for analysts to perform formal analysis from target specification of cryptographic protocols for input as necessary.

R3.3: Supporting tools must make possible for analysts to improve specification of cryptographic protocols for input as necessary.

R3.4: Supporting tools must make possible for analysts to delete specification of cryptographic protocols for input as necessary.

It can be assumed that analysts improve a target cryptographic protocol from the result of formal analysis. Therefore, supporting tools must satisfy **R3** to make tasks of improvement from the result of formal analysis easy for analysts.

R4: Supporting tools must systematically store and manage the result of formal analysis.

R4.1: Supporting tools must store and manage the result of formal analysis for each users.

R4.2: Supporting tools must store and manage the result of formal analysis for each protocols.

R4.3: Supporting tools must store and manage the result of formal analysis for each times of formal analysis.

R4.4: Supporting tools must store and manage the result of formal analysis for each divided cases in tasks of forward reasoning.

It is possible that analysts perform formal analysis of many cryptographic protocols. As the result, it is time-consuming task for analysts to search only a target result of formal analysis of cryptographic protocol in many of those. Therefore, supporting tools must satisfy **R4** to support many cryptographic protocols by analysts.

R5: Supporting tools must make possible for analysts to use the result of formal analysis.

R5.1: Supporting tools must make possible for analysts to read the result of formal analysis.

R5.2: Supporting tools must make possible for analysts to edit the result of formal analysis.

R5.3: Supporting tools must make possible for analysts to delete the result of formal analysis.

It can be assumed that analysts perform analyze from deduced many logical formulas, and manage the many result of formal analysis. Therefore, supporting tools must satisfy **R5** to make these tasks easy for analysts.

6.2 Overview of Supporting Tools

Overview of supporting tools is represented in figure 6.1. At first, analysts input a specification of cryptographic protocol as a target of formal analysis, and obtain the result of formalization. After that, analysts input the result of formalization, and forward reasoning is performed from the result of formalization as premises by FreeEnCal [9]. After that, if it is necessary to classify in case that an intruder or a participant receives data, analysts add logical formulas depending on each case, and input these results as premises for forward reasoning. If forward reasoning in each case, analysts input the result of forward reasoning, and obtain the result whether attack succeed or not in the target cryptographic protocol. If attack succeeds in the target cryptographic protocol, analysts improve the protocol, and repeat formal analysis. By these activities, analysts can concentrate on improving cryptographic protocols [34].

List of supporting tools that makes tasks of formal analysis by the proposed method easy for analysts is as follows.

T1: Formalization tool from specification of cryptographic protocol

T2: Input files making tool depending on branches in task of forward reasoning.

T3: Flaws detection tool from the result of forward reasoning

T1, **T2**, and **T2**, satisfies **R1** in requirement analysis of section 6.1.

List of functions that supports formal analysis by the proposed method easy for analysts is as follows. These functions satisfy the requirement of supporting tools, from **R2** to **R5**.

F1: Specification managing function (satisfied **R2**)

F1.1: Creating new file

F1.2: Saving to corresponding directory

F1.3: Editing saving directory

F1.4: Deleting file

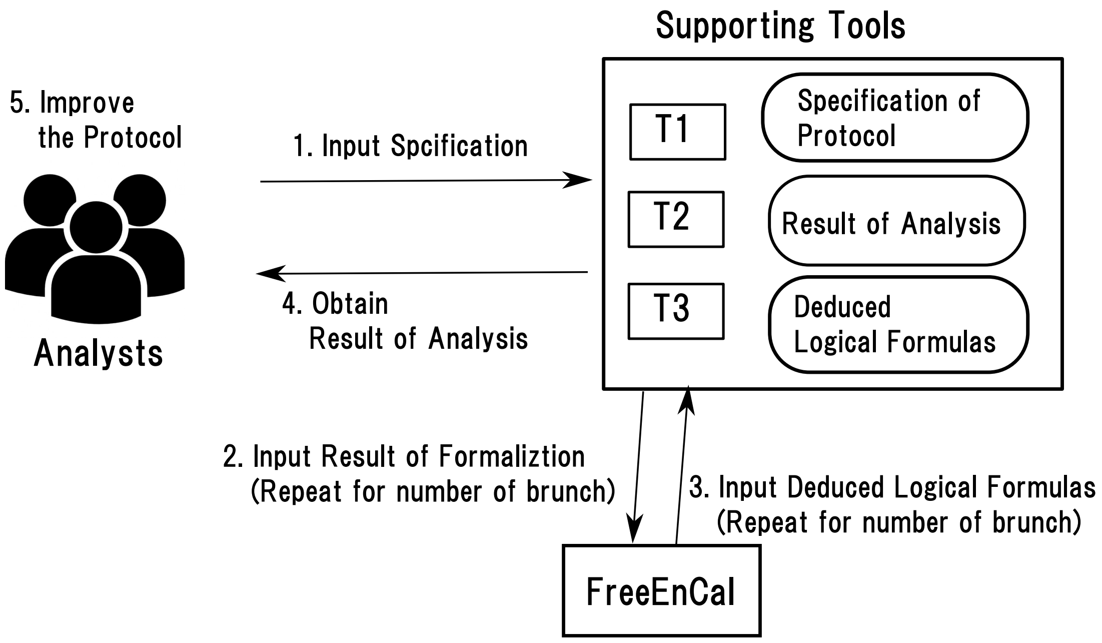

Supporting Tools
5. Improve the Protocol
1. Input Spcification
T1
Specification of Protocol
T2
Result of Analysis
T3
Deduced Logical Formulas
4. Obtain Result of Analysis
Analysts
2. Input Result of Formaliztion (Repeat for number of brunch)
3. Input Deduced Logical Formulas (Repeat for number of brunch)
FreeEnCal

Figure 6.1: Overview of supporting tools.

F2: Making directory function for each user (satisfied **R2**, **R4**)

F3: Making directory function for each target protocol (satisfied **R2**, **R4**)

F4: Extracting specification of cryptographic protocol function in stored files (satisfied **R3**)

F5: Performing formal analysis of extracted specification function (satisfied **R3**)

F6: Extracting result of analysis function (satisfied **R5**)

- **F6.1:** Extracting result of deduced logical formulas
- **F6.2:** Extracting result of analysis that is represented as form of specification

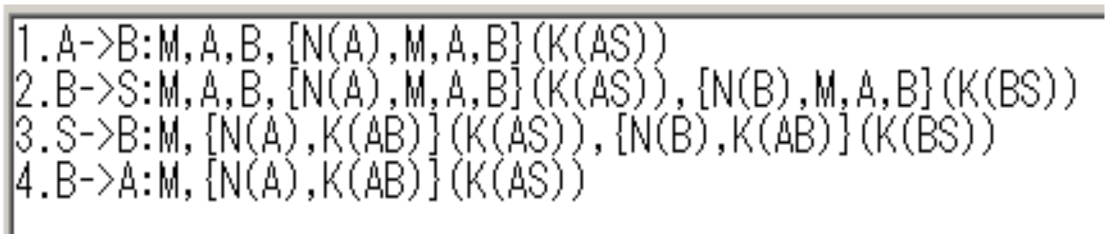
6.3 Implementation of Supporting Tools

I implemented those supporting tools that input file of specification or commands on command line, and output the result of formalization, forward reasoning, and analysis. Programming languages that I used for implementation are Ruby and Perl.

At first, user previously makes the specification by txt file based on eq. 3.1 in section 3.1. Concretely, input file is represented as following form.

```
N.X1->X2:Y1,Y2,...,Yn
```

In the form, N is the number of step. X1 and X2 are participants. Y1,Y2,...,Yn are sent and received data. If encrypted data is sent, it is represented as { Y1,Y2,...,Yn }. If data



```
1.A->B:M,A,B,{N(A),M,A,B}(K(AS))
2.B->S:M,A,B,{N(A),M,A,B}(K(AS)),{N(B),M,A,B}(K(BS))
3.S->B:M,{N(A),K(AB)}(K(AS)),{N(B),K(AB)}(K(BS))
4.B->A:M,{N(A),K(AB)}(K(AS))
```

Figure 6.2: Example of input for supporting tools (target of formal analysis).

that a participant owns is sent, it is represented as "(X)" (X is a participant). Figure 6.2 shows example of input file that is represented as the form.

After the input file is made, user inputs following command.

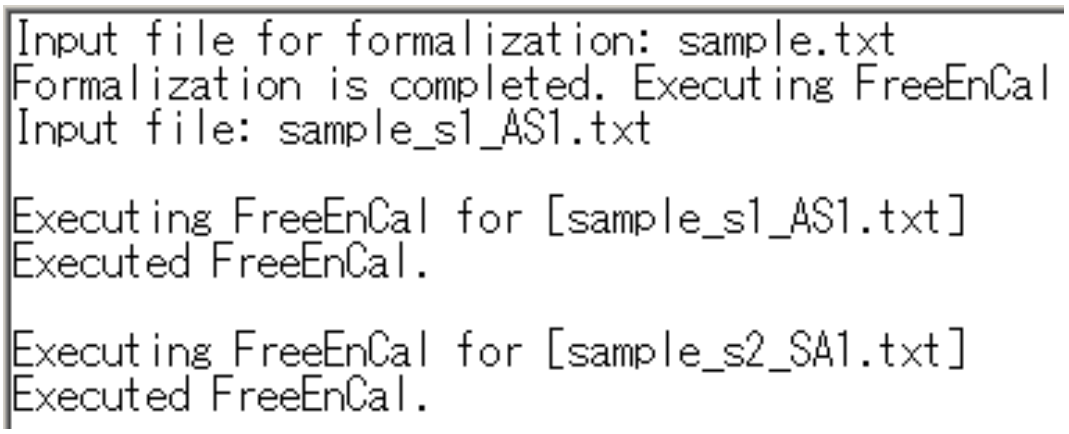
```
ruby analysistool.rb
```

After that, user input specification of cryptographic protocol by txt file. As the result, formalization of the input specification is performed automatically. User obtains the result of formalization by txt file that is represented by logical formulas and form of FreeEnCal.

After the task of formalization, forward reasoning from the result of formalization as premises by FreeEnCal is performed automatically. User obtains output file as the result of forward reasoning in each case that is represented by logical formulas. Logical formulas that represent a participant or intruder sends data are added in the output file, and forward reasoning by using the is repeated until any logical formulas cannot be added. The name of output file is represented following form.

```
[Filename]_s[Step number]_[Sender][Receiver][File number].txt
```

Figure 6.3 shows the example of formalization and forward reasoning by using supporting tools.



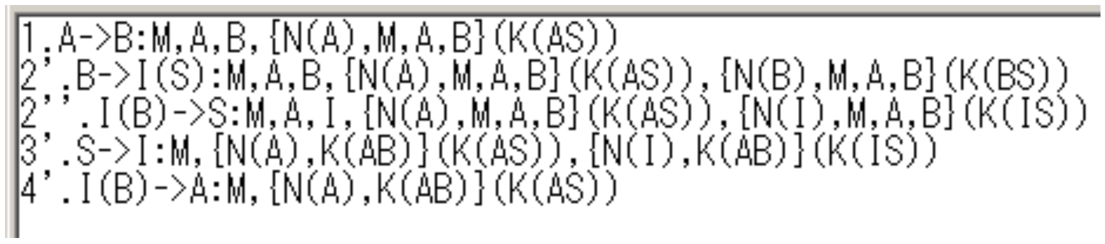
```
Input file for formalization: sample.txt
Formalization is completed. Executing FreeEnCal
Input file: sample_s1_AS1.txt

Executing FreeEnCal for [sample_s1_AS1.txt]
Executed FreeEnCal.

Executing FreeEnCal for [sample_s2_SA1.txt]
Executed FreeEnCal.
```

Figure 6.3: Formalization and forward reasoning by using supporting tools.

If a result of forward reasoning represents success of attack, “Flaw” is included in name of output file. On the other hand, user obtains list of actions by participants and an intruder in each case. The list of actions is translated into form of specification that is represented as figure 6.4. As the result of translation, user can know how an intruder succeeded the attack.



```
1.A->B:M,A,B,{N(A),M,A,B}(K(AS))
2'.B->I(S):M,A,B,{N(A),M,A,B}(K(AS)),{N(B),M,A,B}(K(BS))
2''.I(B)->S:M,A,I,{N(A),M,A,B}(K(AS)),{N(I),M,A,B}(K(IS))
3'.S->I:M,{N(A),K(AB)}(K(AS)),{N(I),K(AB)}(K(IS))
4'.I(B)->A:M,{N(A),K(AB)}(K(AS))
```

Figure 6.4: Example of attacks by form of specification.

6.4 Evaluation of Supporting Tools

I show that implemented supporting tools can reduce duration in performing formal analysis by the proposed method compared to perform manually. The proposed method is divided into 3 tasks, formalization for each branch, forward reasoning for each branch, and detecting fatal actions of attack from deduced logical formulas by forward reasoning. I compare duration of these tasks in case of performing manually and using the supporting tool, and demonstrate that the duration can be reduced by the supporting tool.

I demonstrated that duration of formalization was reduced by using implemented supporting tools. It is time-consuming task for analysts to perform formalization manually [34]. Table 6.1 shows duration of formalization for Second Protocol Attempt [5], Third Protocol Attempt [5], Needham-Schroeder Shard Key Protocol [24], and Otway-Rees Protocol [27]. As shown in table 6.1, it took about 15 minutes to formalize manually. I performed formalization of these protocols by using implemented supporting tools, formalization could be finished automatically and instantly. Therefore, implemented supporting tools can reduce duration of formalization.

Kind of Protocol	Duration Manually
Second Protocol Attempt	13 minutes
Third Protocol Attempt	13 minutes
Needham-Schroeder Shard Key	20 minutes
Otway-Rees	25 minutes

Table 6.1: Duration of formalization manually

It was demonstrated that duration of forward reasoning was reduced by using implemented supporting tools. It is time-consuming task for analysts to make input file of forward reasoning by using FreeEnCal for all cases that an intruder or a participant sends falsified data. Furthermore, command for executing FreeEnCal is complex as follows.

```
$ FreeEnCal -s -t [output file] -f CSV [input file] >& output.log
```

Therefore, it is also time-consuming task for analysts to execute FreeEnCal for those all cases. It takes about one hour to perform make input file and execute FreeEnCal manually for those all cases. I performed those tasks automatically by using implemented supporting tool. Therefore, implemented supporting tools can reduce duration of forward reasoning.

I demonstrated that duration of analysis was reduced by using implemented supporting tools. It is time-consuming task for analysts to detect fatal actions of attacks in deduced many logical formulas by forward reasoning. Furthermore, it is time-consuming task for analysts to judge success of attack or not from detected actions manually as shown in figure 6.2 [34]. I performed those tasks for those protocols by using implemented supporting tool. As the result those tasks could be finished automatically and instantly. Therefore, implemented supporting tools can reduce duration of analysis.

Kind of Protocol	**Duration Manually**
Second Protocol Attempt	3 minutes
Third Protocol Attempt	3 minutes
Needham-Schroeder Shard Key	5 minutes
Otway-Rees	6 minutes

Table 6.2: Duration of judging success of attack or not from detected actions manually

By implemented supporting tools, analysts can perform the proposed method in a short time. Furthermore, it can be assumed that analysts perform the proposed method for many cryptographic protocols. It is hopeful that analysts can concentrate on improving and designing of protocol.

Whole duration of the proposed method by using supporting tools is about several hours. It can be said that the duration is practical time because it takes very long time for designing and improving of cryptographic protocols. Proposed and practically used cryptographic protocols should be carefully analyzed and improved with taking time. For example, in case of SSL protocol [21], it takes 1 year to improve version 2.0 into version 3.0 [19]. Furthermore, in case of TLS protocol [21], it takes 7 years to improve version 1.0 [2] into version 1.1 [15], and takes 2 years to improve version 1.1 into version 1.2 [16].

Chapter 7

Discussion

The proposed method has limitation that completeness cannot be said. The limitation is following **L1**, **L2**, and **L3**.

L1: Target fatal actions of attacks that can be detected is depending on an intruder model.

In the proposed method, behavior of an intruder is based on Dolev-Yao model [17]. This model assumes that an intruder can eavesdrop, falsify, decryption by a key that an intruder has. The proposed method can systematically detect fatal actions of attacks that are based on these attacks. However, Dolev-Yao model cannot support to detect fatal actions of attacks about mathematical computation. For example, in case of coin-flips protocol that represents in section 5.3, if an intruder can estimate secret data x from get data $f(x)$ by mathematical computation, an intruder can success attack in the protocol [31]. However, fatal actions that represent the attack could not be detected in 5.3.

L2: Deduced logical formulas are limited in using FreeEnCal.

In using FreeEnCal, degree of nested logical connectives must be set in order to limit the range of deduced logical formulas [9]. If there are fatal actions of attacks outside of the range, those fatal actions cannot be detected.

L3: It is limited to enumerate falsified data that an intruder can send.

In the current proposed method, it is considered what data is falsified or not, but it is not considered how to falsify each data. For example, in case of Otway-Rees protocol that is represented in section 3.5, the current proposed method cannot detect fatal actions of attacks that represents success of attack to second flaws because falsified data about second flaws cannot generated by the current proposed method.

Those limitation **L1**, **L2**, and **L3** can be reduced by extending the proposed method. About **L1**, fatal actions of attacks about mathematical computation can be detected by adding logical formulas that represent “a participant gets another data by calculation for collected data in a protocol” in section 4.2.4. However, it must be cleared what new data can get and what data should be collected to get the new data. About **L2**, fatal actions of attacks outside of the range can be detected by setting high degree in using FreeEnCal. However, if the set degree is very high, it is possible that forward reasoning by using FreeEnCal cannot be stopped or it takes very long time for forward reasoning. About **L3**, those fatal actions of attacks can be detected by extending method of enumerating data with falsifying that an intruder can send. However, if the number of enumerated falsified data is too much by the extension, it is possible that the number of branches is very increased and the proposed method cannot be stopped. Therefore, it is future work how to extend the proposed method without those problems.

Soundness can be said by the result of detecting fatal actions of attacks in the proposed method. In the proposed method, if actions of an intruder that represent falsification or eavesdropping are included, a person’s action in the last step of protocol is detected as a fatal action of attack. If a person’s action in last step of protocol is detected, following two cases are possible. At first, an intruder does not perform actions of falsification or eavesdropping in a cryptographic protocol. Second, participants cannot recognize actions of falsification or eavesdropping by an intruder. If an intruder falsified a target cryptographic protocol, deduced logical formulas must include $Send(i, p, x)$ and $Recv(i, x')$ (p is any participant, x is any sent data, and x' is falsified data). $Send(i, p, x)$ and $Recv(i, x')$ are generated by tasks of enumerating falsified data in section 4.2.5. In the proposed method, it is assumed that an intruder does not forge sending data if participants can recognize falsification. Therefore, if $Send(i, p, x)$, $Recv(i, x')$ and a person’s action in last step of protocol are detected in deduced logical formulas, it represents falsification by an intruder. On the other hand, if an intruder eavesdropped a target cryptographic protocol, deduced logical formulas must include $Get(i, sdata)$ ($sdata$ is an individual constant that represents secret data). $Get(i, sdata)$ is deduced in behavior of an intruder that represents “if a participant sends x, an intruder gets it” in section 4.2.3. Eavesdropping cannot be recognized by participants because an intruder only gets sent data, and does not falsify sent data. Therefore, if a person’s action in last step of protocol and $Get(i, sdata)$ are detected, it represents eavesdropping by an intruder.

Chapter 8

Conclusions

8.1 Contributions

I showed that the proposed method can be applied to those 19 cryptographic protocols in figure 4.1 of section 4.1. At first, I proposed formal analysis method with reasoning for key exchange protocols. After that, I compared 19 cryptographic protocols based on participants' behavior and the number of participants, and five differences among 19 cryptographic protocols were found as I explained in figure 4.1 of section 4.1. After that, I extended the proposed method in order to represent those five differences. By the extended method, I performed formal analysis of some cryptographic protocols that include five differences among cryptographic protocols. As the result, I could re-detect fatal actions of attacks that represent success of attack that is pointed out or I could perform tasks of the proposed method to the end.

As the result of evaluation of implemented supporting tools, I showed that analysts can reduce duration of the proposed method by using those supporting tools. Therefore, it can be said that analysts can easily perform the proposed method by implemented supporting tools.

8.2 Future Works

In the future, at first, the proposed method should be extended in order to be applied to more target attacks. In current proposed method, target fatal actions of attacks for detecting and falsified data that an intruder can send are limited. However, if the proposed method is extended without those limitations, there can be a problem that tasks of the extended method cannot be finished because tasks of the extended method are increased, especially falsified data that an intruder can send. Therefore, the proposed method must be extended without the problem.

Second, the implemented supporting tools should be made more easy to use for analysts. In current supporting tools, those output files are stored in same directory. In order to manage and store those output files more systematically, it is necessary to classify those output files, for example, execution date, forward reasoning about each step. On the other hand, input file for formal analysis is made manually. Therefore, it is necessary to implement a supporting tool for making input file.

Finally, the proposed method should be applied to various cryptographic protocols. I showed that the proposed method can be applied to formal analysis of cryptographic protocols. It is necessary to evaluate security of target cryptographic protocol by the proposed method.

Publications

Refereed papers published in journals or books (first author)

- Kazunori WAGATSUMA, Shogo ANZE, Yuichi GOTO, and Jingde CHENG: Formalization for Formal Analysis of Cryptographic Protocols with Reasoning Approach, in J. J. Park, Y. Pan, C.-S. Kim, and Y. Yan (Eds.), "Future Information Technology, FutureTech 2014," Lecture Notes in Electrical Engineering, Vol. 309, pp. 211-218, Springer, Heidelberg, May 2014.
- 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).
- 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, in J. J. Park, H. Chao, H. Arabnia, and N. Y. Yen (Eds.), "Advanced Multimedia and Ubiquitous Engineering - Future Information Technology," Lecture Notes in Electrical Engineering, Vol. 354, pp. 25-32, Springer, Heidelberg, July 2015.

Refereed papers published in journals or books (co-author)

- Kai SHI, Kazunori WAGATSUMA, Yuichi GOTO, and Jingde CHENG: World Model, Predictive Model, and Behavioral Model of an Anticipatory Reasoning-Reacting System for Runway Incursion Prevention, International Journal of Computing Anticipatory Systems, Vol. 28, pp. 67-88, CHAOS, December, 2014.
- Shunsuke NANAUMI, Kazunori WAGATSUMA, Hongbiao GAO, Yuichi GOTO, and Jingde CHENG: A Bidirectional Transformation Supporting Tool for Formalization with Logical Formulas, in N. T. Nguyen, et al. (Eds.), "Intelligent Information and Database Systems, 7th Asian Conference, ACIIDS 2015, Bali, Indonesia, March 23-25, 2015, Proceedings," Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science), Vol. 9011, pp. 634-643, Springer, March 2015.
- Shunsuke NANAUMI, Kazunori WAGATSUMA, Hongbiao GAO, Yuichi GOTO, and Jingde CHENG: Development of a Bidirectional Transformation Supporting Tool for Formalization with Logical Formulas and Its Application, in J. J. Park, H. Chao, H. Arabnia, and N. Y. Yen (Eds.), "Advanced

Multimedia and Ubiquitous Engineering - Future Information Technology," Lecture Notes in Electrical Engineering, Vol. 352, pp. 1-6, Springer, Heidelberg, May 2015.

- Da BAO, Kazunori WAGATSUMA, Hongbiao GAO, and Jingde CHENG: Predicting New Attacks: A Case Study in Security Analysis of Cryptographic Protocols, Lecture Notes in Electrical Engineering, Springer, Heidelberg (accepted).

Refereed papers published in international conference proceedings (first author)

- Kazunori WAGATSUMA, Yuichi GOTO, and Jingde CHENG: Formal Analysis of Cryptographic Protocols by Reasoning Based on Deontic Relevant Logic: A Case Study in Needham-Schroeder Shared-Key Protocol, Proc. 11th International Conference on Machine Learning and Cybernetics, pp. 1866-1871, Xian, China, The IEEE Systems, Man, and Cybernetics Society, July 2012.

Refereed papers published in international conference proceedings (co-author)

- Shunsuke NANAUMI, Kazunori WAGATSUMA, Yuichi GOTO, and Jingde CHENG: Development of a Supporting Tool for Translation between Declarative Sentences and Logical Formulas, Proc. 12th International Conference on Machine Learning and Cybernetics, pp. 1179-1184, Tianjin, China, The IEEE Systems, Man, and Cybernetics Society, July 2013.

Bibliography

[1] Martín ABADI and Cédric FOURNET: Mobile Values, New Names, and Secure Communication, Proceedings of the 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pp. 104-115, ACM, March 2001.

[2] Christopher ALLEN and Tim DIERKS: The TLS Protocol Version 1.0, http://tools.ietf.org/html/rfc2246.txt, January 1999.

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

[4] 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.

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

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

[7] 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.

[8] Jingde CHENG and Junichi MIURA: Deontic Relevant Logic as the Logical Basis for Dpecifying, 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.

[9] 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.

[10] John CLARK and Jeremy JACOB: A Survey of Authentication Protocol Literature: Version 1.0, November 1997.

[11] Cas JF 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.

[12] Cas JF CREMERS: On the protocol composition logic PCL, Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security, ACM, pp. 66-76, March 2008.

[13] Dorothy E. DENNING and Giovanni Maria SACCO: Timestamps in Key Distribution Protocols, Communications of the ACM, Vol. 24, No. 8, pp. 533-536, August 1981.

[14] Razvan DIACONESCU and Kokichi FUTATSUGI: Cafeobj Report, World Scientific, 1998.

[15] Tim DIERKS and Eric RESCORLA: The Transport Layer Security (TLS) Protocol Version 1.1, https://tools.ietf.org/html/rfc4346.txt, April 2006.

[16] Tim DIERKS and Eric RESCORLA: The Transport Layer Security (TLS) Protocol Version 1.2, http://www.ietf.org/rfc/rfc5246.txt, August 2008.

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

[18] Santiago ESCOBAR, Catherine MEADOWS, and Jose MESEGUER: Maude-NPA: Cryptographic Protocol Analysis Modulo Equational Properties, Lecture Notes in Computer Science, Vol. 5705, pp. 1-50, Springer-Verlag, June 2009.

[19] Alan FREIER: The SSL Protocol Version 3.0, http://wp.netscape.com/eng/ssl3/draft302.txt, November 1996.

[20] Hongbiao GAO, Yuichi GOTO, and Jingde CHENG: A Systematic Methodology for Automated Theorem Finding, Theoretical Computer Science, Vol. 554, pp. 2-21, Elsevier, October 2014.

[21] Kipp HICKMAN and Taher ELGAMAL: The SSL protocol, Netscape Communications Corporation, June 1995.

[22] ISO, ISO/IEC 29128: Information Technology – Security Techniques – Verification of Cryptographic Protocols, 2011.

[23] Sebastian MODERSHEIM and Luca VIGANO: The Open-Source Fixed-Point Model Checker for Symbolic Analysis of Security Protocols, Lecture Notes in Computer Science, Vol. 5705, pp. 166-194, Springer-Verlag, June 2009.

[24] Roger NEEDHAM and Michael D. SCHROEDER: Using Encryption for Authentication in Large Networks of Computers, Communication of the ACM, Vol. 21, No. 12, pp. 993-999, December 1978.

[25] Monica NESI and Giustina NOCERA: Deriving the Type Flaw Attacks in the Otway-Rees Protocol by Rewriting, Nordic Journal of Computing, Vol. 13, No. 1, pp. 78-97, June 2006.

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

[27] Dave OTWAY and Owen REES: Efficient and Timely Mutual Authentication, ACM SIGOPS Operating Systems Review, Vol. 21, No. 1, pp. 8-10, ACM, January 1987.

[28] Lawrence C. PAULSON: Isabelle: A Generic Theorem Prover, Lecture Notes in Computer Science, Vol. 828, Springer, July 1994.

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

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

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

[32] Kazunori WAGATSUMA, Shogo ANZE, Yuichi GOTO, and Jingde CHENG: Formalization for Formal Analysis of Cryptographic Protocols with Reasoning Approach, Future Information Technology, Lecture Notes in Electrical Engineering, Vol. 309, pp. 211-218, Springer, Heidelberg, May 2014.

[33] 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).

[34] 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.