MODELING AND ANALYSIS OF TWO-FACTOR AUTHENTICATION PROTOCOL FOR USB DIGITAL EVIDENCE ACQUISITION DEVICES

SIRIPOOM LAPTIKULTHAM

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR
THE DEGREE OF MASTER OF ENGINEERING
(COMPUTER ENGINEERING)
FACULTY OF GRADUATE STUDIES
MAHIDOL UNIVERSITY
2015

Thesis
entitled

MODELING AND ANALYSIS OF TWO-FACTOR AUTHENTICATION PROTOCOL FOR USB DIGITAL EVIDENCE ACQUISITION DEVICES

……..

Mr. Siripoom Laptikultham
Candidate

……..

Asst. Prof. Suratose Tritilanunt,
Ph.D. (Information Technology)
Major advisor

……..

Asst. Prof. Konglit Hunchangsith,
Ph.D. (Information Technology)
Co- advisor

……..

Lect. Noppadol Wanichworanant,
Ph.D. (Electrical Engineering)
Co- advisor

……..

Prof. Patcharee Lertrit,
M.D., Ph.D. (Biochemistry)
Dean
Faculty of Graduate Studies
Mahidol University

……..

Assoc. Prof. Rangsipan Marukatat,
Ph.D. (Computer Science)
Program Director
Master of Engineering Program in
Computer Engineering
Faculty of Engineering
Mahidol University

Thesis
entitled

MODELING AND ANALYSIS OF TWO-FACTOR AUTHENTICATION PROTOCOL FOR USB DIGITAL EVIDENCE ACQUISITION DEVICES

was submitted to the Faculty of Graduate Studies, Mahidol University
for the degree of Master of Engineering
(Computer Engineering)
on
June 2, 2015

……..
Mr. Siripoom Laptikultham
Candidate

……..
Lect. Narit Hnoohom,
Ph.D. (Computer Engineering)
Chair

……..
Asst. Prof. Suratose Tritilanunt,
Ph.D. (Information Technology)
Member

……..
Asst. Prof. Konglit Hunchangsith,
Ph.D. (Information Technology)
Member

……..
Asst. Prof. Supakorn Kungpisdan,
Ph.D. (Computer Science and Software Engineering)
Member

……..
Lect. Noppadol Wanichworanant,
Ph.D. (Electrical Engineering)
Member

……..
Prof. Patcharee Lertrit,
M.D., Ph.D. (Biochemistry)
Dean
Faculty of Graduate Studies
Mahidol University

……..
Lect. Worawit Israngkul,
M.S. (Technical Management)
Dean
Faculty of Engineering
Mahidol University

ACKNOWLEDGEMENTS

First of all, I wish to express my sincere thanks to my major advisor Asst. Prof. Suratose Tritilanunt , and co-advisor Asst. Prof. Konglit Hunchangsith, Lect. Noppadol Wanichworanant, for their valuable advice guidance, kindness support, encouragement and improving this research

My special thanks are sincerely to Lect. Narit Hnoohom, the committee chair and Asst. Prof. Supakorn Kungpisdan, the external examiner of thesis defense for their kindness, attentiveness and time sacrifice for this research.

I take this opportunity to record my sincere thanks to all the lectures and staff of Computer Engineering Program, Faculty of Engineering, Mahidol University for their help and kindness support.

I also place on record, my sense of gratitude to one and all who, directly or indirectly, have lent their helping hand on this thesis.

Last but not least, I desire to express my deeply gratitude to my family, my friends and my colleagues for kindness support, encouragement, and understanding.

Siripoom Laptikultham

MODELING AND ANALYSIS OF TWO-FACTOR AUTHENTICATION PROTOCOL FOR USB DIGITAL EVIDENCE ACQUISITION DEVICES

SIRIPOOM LAPTIKULTHAM 5438250 EGCO/M

M.Eng.(COMPUTER ENGINEERING)

THESIS ADVISORY COMMITTEE : SURATOSE TRITILANUNT, Ph.D., KONGLIT HUNCHANGSITH, Ph.D., NOPPADOL WANICHWORANANT, Ph.D.

ABSTRACT

Currently, there are several problems regarding the usage of a USB storage device. One major concern is in the problem of lacking authorization and in the authentication of software and data stored on that storage devices. Many research papers offer high-levels of security by adding *Two-factor authentication* to USB devices and use additional equipment such as Smartphones, or Token-Based devices, to generate this parameter. These devices help the user to create a second parameter of authentication such as OTP (One-time password) or PIN (Personal identification number). However, these protocols are still inappropriate in terms of adopting into devices for digital forensic applications.

This thesis reviews some related literatures of authentication protocols. We developed the protocol for USB authentication, which was considered to focus on light-weight computation and require the minimum number of exchanged messages between 2 entities. Then the proposed model was analyzed by using a Coloured Petri Net, which is the Formal method tool for modeling and verifying the communication system. We also created two difference scenarios of attacker in the model in order to show that our purposed protocol had been proven secure. Finally, the prototype software of the proposed protocol is implemented thus validating the protocol functionality and workflow.

KEY WORDS: TWO-FACTOR AUTHENTICATION / COLOURED PETRI NET MODEL / CPN TOOLS / USB DATA STORAGE DEVICE

71 pages

การสร้างแบบจำลองและการวิเคราะห์โปรโตคอลยืนยันตัวตนแบบสองระดับสำหรับอุปกรณ์เก็บข้อมูลหลักฐานทางดิจิทัลแบบ USB

MODELING AND ANALYSIS OF TWO-FACTOR AUTHENTICATION PROTOCOL FOR USB DIGITAL EVIDENCE ACQUISITION DEVICES

ศิริภูมิ ลัพธิกุลธรรม 5438250 EGCO/M

วศ.ม.(วิศวกรรมคอมพิวเตอร์)

คณะกรรมการที่ปรึกษาสารนิพนธ์: สุรทศ ไตรติลานันท์, Ph.D.,คงฤทธิ์ หันจางสิทธิ์, Ph.D.,นภดล วณิชวรนันท์, Ph.D.

บทคัดย่อ

ปัจจุบันมีปัญหาหลายประการที่เกี่ยวกับการใช้งานอุปกรณ์เก็บข้อมูล USB ซึ่งหนึ่งในนั้นคือการที่ปราศจากการอนุญาตให้ใช้งานและการยืนยันตัวตนของซอฟแวร์และข้อมูลที่ถูกเก็บไว้ภายในอุปกรณ์นั้นๆ มีงานวิจัยหลายฉบับที่นำเสนอระบบความปลอดภัยที่สูงขึ้นโดยการใช้ *การยืนยันตัวตนโดยใช้ 2 ปัจจัย* กับอุปกรณ์หน่วยความจำแบบต่อพ่วงชนิด USB โดยมีอุปกรณ์เสริมอย่างเช่น สมาร์ทโฟน หรือ อุปกรณ์ที่เป็น Token-based ซึ่งสามารถสร้างสถานะสำหรับการยืนยันตัวตนขั้นที่สองได้ เช่นการใช้รหัสผ่านแบบใช้ครั้งเดียวหรือรหัสยืนยันส่วนบุคคล แต่โปรโตคอลเหล่านี้ยังไม่เหมาะสมที่จะนำมาใช้กับอุปกรณ์ที่พัฒนาขึ้นมาเพื่อใช้สำหรับงานทางด้านนิติวิทยาดิจิทัล

งานวิจัยฉบับนี้ผู้วิจัยได้ศึกษางานวิจัยอื่นที่เกี่ยวข้องกับโปรโตคอลการยืนยันตัวตนและได้ทำการพัฒนาโปรโตคอลการยืนยันตัวตนสำหรับอุปกรณ์เก็บข้อมูลชนิด USB โดยเน้นไปที่การประมวลผลต่ำและจำนวนขั้นตอนในการแลกเปลี่ยนข้อมูลน้อย โดยโปรโตคอลจะถูกวิเคราะห์ด้วย Coloured Petri Net ซึ่งเป็นเครื่องมือในการสร้างและทดสอบระบบ โดยจำลองผู้โจมตีระบบขึ้น 2 สถานการณ์เพื่อแสดงให้เห็นว่าโปรโตคอลการยืนยันตัวตนที่ถูกพัฒนาขึ้นนั้นมีความปลอดภัย นอกจากนั้นผู้วิจัยได้พัฒนาโปรแกรมต้นแบบของกระบวนการการยืนยันตัวตนขึ้น เพื่อตรวจสอบความถูกต้องในการทำงานของระบบอีกด้วย

71 หน้า

CONTENTS

CONTENTS (cont.)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (cont.)

LIST OF FIGURES (cont.)

CHAPTER I
INTRODUCTION

1.1 Background and State of Problems

The digital evidence acquisition is the most important process for the investigation procedure in digital forensics field, because the essential data can be analyzed and extracted. The vital data, which is stored in *volatile memory* such as RAM (Random access memory); are consists of running processes and applications, network ports and connections, cache files, and attached device list. Due to the sensitivity of the volatile memory, the stored data can be deleted or altered by replacing them with the new running process. In order to acquire this sensitive data, the investigator needs to minimize the process on the target computer. From this reason, the digital evidence acquisition devices have to processes as least as possible to acquire the sensitive evidence from volatile memory in order to avoid the data altering.

Another important consideration is when the device has been stolen or used by unauthorized client; the data can be accessed and misused without protection. The acquired data might be altered and lost its correctness and integrity. Moreover, an unauthorized user can duplicate the digital evidence acquisition software installed onthe device and use for one's own sake. In these days, most of commercial Universal Serial Bus (USB) digital evidence acquisition devices do not have any data encryption function or user authentication process, as well as the feature to prevent the device from any unauthorized usages. Using only Username and Password to authenticate the user is not secure enough, because the computer with high-computation power is not too expensive and be able to crack the predictable password in a couple of weeks. Many studies suggest adding an additional layer of security to the devices by using additional equipments such as Smartphone or Token-based device. These devices can generate the second factor which will be used in the two-factor authentication protocol

such as OTP (One-time password) or PIN (Personal identification number). Many proposed protocols, two-factor authentication have been proven that they are more secure than the normal authentication. However, these protocols are still inappropriate to applied to the digital evidence acquisition devices for used in digital forensic applications.

1.2 Objectives

1) To research and compare among several two-factor authentication protocols.

2) To develop an appropriated authentication protocol for USB type data storage.

3) To propose the authentication protocol that can prevent access from unauthorized users.

4) To propose the authentication protocol that can prevent the device from abused in an unassigned task and limit the access time.

5) To propose the authentication protocol that can prevent duplication of software from the device to others.

6) To verify the security of purposed protocol by using formal verification tool.

7) To validate the functionality of the proposed protocol by implementing prototype software.

1.3 Scope of Research

1) The proposed authentication protocol be proven and verified the correctness in process using formal verification tool.

2) The proposed authentication protocol is able to apply to USB storage devices in order to use them as a digital evidence collecting tool.

3) The proposed authentication protocol is able to protect data duplication from the mass storage device to the others.

4) The proposed authentication protocol is able to use on any type of external storage devices.

5) The proposed authentication protocol is able to prevent from an unauthorized access.

6) The proposed authentication protocol is able to limit the number of access time.

1.4 Results

1) Understanding the protocol designing process along with advantages and disadvantages of each protocol in order to design the proposed protocol.

2) The proposed authentication protocol can apply to any USB type data storage devices.

3) The unauthorized user cannot access the data stored in the device.

4) The proposed authentication can prevent the device from abuse and limit the number usage.

5) The proposed authentication can prevent the user from unauthorized usage, after duplicating the software and data from the device to others.

6) The proposed authentication protocol is proven to be secured and verified the correctness using formal verification tool.

7) The functionality of the prototype software is proven to be valid.

CHAPTER II
LITERATURE REVIEW

In this section, the literatures and research papers which are related to the authentication protocol, especially designed for USB-type external data storage devices, will be reviewed. Then, we will review tools for validating and verifying the authentication protocol in term of functionality and security. In order to design and implement the protocol and apply for use in digital forensic applications.

2.1 Authentication Protocol for Mass Storage

In this section, the literatures related to the authentication protocol will be reviewed. This research will focus on the authentication protocols which were designed for using with USB data storage device. This well-known mass storage device is widely used because of its convenient but lack of security. A lot of research suggest the way to add security to the device which is the authentication protocol, used with other devices i.e., Smartphone or token which contains secret data for use during the authentication and verification process.

The first paper Kyungroul Lee et al [1] they show many types of authentication protocol for USB type of data storage device and consider their exploits. Then, they classified the exploits into 12 categories based-on the authentication protocols and demonstrated for each type of vulnerable. The research also divided commercial products into 3 groups. The first group is software-only approach, which the host-device had to memorized user ID and password of the device without encryption in every transaction. Then the important data were stored into the memory in plain-text form. Due to the transaction protection, registration and authentication process should be done during offline. The second group of commercial products is hardware-support partition approach; this type divided the memory storage

into partitions. One of the partitions is initially installed software for managing the authentication and encryption other partitions. There is not only the usage of private session key in the registration phase and public key in the authentication phase, but it also stores username and password in plain-text form. If the transactions had been sniffed, then the message could be compromised by the brute-force attack and the reverse engineering attack to seek for the relation between the session key and the encrypted username and password. Lastly, the hardware-based approach, with the major characteristic, uses offline registration and challenge-response in authentication process. The registration process should have been done in both offline and online to create two shared keys for the authentication process. During the authentication process, host (server) has to generate random nonce for verify with USB data storage device. This approach has more security level than others. But there is some vulnerable flaw in the verification process because it is difficult to protect transactions from sniffing. Response message in the authentication process could be used for replay attack; however using one-time password is an acceptable solution. Besides, minimize or maximize number of transactions are also reasonable too.

This study concludes that all three types of authentication protocols for commercial USB data storage device focus on data encryption algorithm than the authentication protocol. As a result of this, the authentication protocols are vulnerable since attackers can use for attack the process.

Another authentication protocol is from Yuw-Yi Yang et al. [2] research paper. Without other additional devices, user need to fill username and password correctly to gained an access. Otherwise, the transactions between attached USB and host computer are prohibited. There are three characteristics on this scheme. First of all, only verified user can access the data stored in USB. Secondly, if the USB has been stolen, data decryption is impossible without a key. Third, even the legal user is going to distribute the data, but if the server suspended the corresponding key then the stolen USB could not be accessed anymore. The protocol was split into two phases, registration and verification.

In the registration phase, system will block the user from sending packets through USB port. User needs to create username and password and send them to the server for the authentication and generate an encryption key. When user has been

verified through authentication process correctly, the system will allow accessing the USB storage. The data packets will be encrypted using the key before sending to USB data storage device. This scheme has the completeness of data acquisition, because not only it can prevent the data lost but it also needs to pass through the authentication process again if user wants to decrypt the data to access the original file. This scheme can resist an offline password guessing attack because the attacker will never know the random generated session key for each use. The newly nonce will be generated in every single use to resist the replay attack. Even the verifier had been stolen; the attacker has no knowledge about the private session key to verify with server. This authentication protocol needs to communication with server twice only. Therefore, this scheme provides the security effectively for the important data.

The next one is an authentication method for secure data transfer via USB interface in order to maintain both security and efficiency. A. N. Magdum et al. [3] had developed a secure procedure for USB mass storage devices usage. Starting with the registration phase, the user needs to register ID and password at the Authentication Server. In the verification phase, the Authentication Server verifies that user. If that ID and password are valid and user wants to transmit file to USB, the device needs to be accessed by ID and password. In the data encryption-decryption phase, the data which are transmitted via USB interface will be encrypted or decrypted by session key, which is generated from the previous phase. This research proposes an additional security layers to USB storage devices while reducing some convenience to the users because they have to add the step of encryption and decryption process on the data transmission via USB interface. However, the researcher only tests the developed protocol on document files, but the data transfer time possibly will be increased on the large file which will make user needs more time to transmit the file. This research has a similar characteristic in term of an authentication technique which is able to be adopted to and used for our work.

Another paper studied an authentication protocol for mass storage devices by Elderfrawy et al.[4], which uses two factors authentication scheme to enhance the security. The study used a Smartphone as the second factor for authentication process. Only the user who rightly follows the registration procedure is able to access the data stored in the device. The researcher found the vulnerable of Yang et al.[5] proposed

protocol based on server private key impersonation attack, so he developed the new scheme in order to solve the problem. This scheme separates into two phases; the registration phase and verification phase. In the registration phase, the client sends his identities and data storage device identity to the server. Then, the server replies security information back to client's Smartphone. During the verification phase, the client has to input the correct user ID and password when plug the device to the computer. The computer will pass parameters to Smartphone using wireless connection then the Smartphone will send to the Authentication Server. After the Authentication Server verifies the parameters, if the parameter of the Smartphone is valid, the Smartphone will allow the client to access the data storage device. This scheme is able to prevent many basic types of attack such as, offline password guessing and replay attack. Using Smartphone as an additional factor can limit the access of data storage device to only legitimate user. This protocol is appropriate for protecting the sensitive data in business such as classified documents or financial information. Because the proposed protocol will encrypts the data using the client computer before transfers to the USB device. Only legitimated user who has registered and possessed the specified Smartphone can access and decrypts the data in the USB device.

Interesting research, the Three-factor authentication protocol method was proposed by Lee et al. [6]. This research applies all three types of factor to authenticate the user which consists of user account (Username and Password), Smartcard, and User's biometric. However, the protocol needs precisely biometric input to authenticate the user identification. Consequently, He et al. [7] introduced the enhanced version of the protocol with fuzzy extractor function to give the error-tolerance to the biometric factor inputs. Both proposed protocols are proven secured. Nevertheless, these two protocols are appropriate for business usage due to the need of special device for smartcard and biometric inputs. The client's computer also needs to communicate with the Authentication Server for verification, then encrypts the data before transferring it to USB device.

Then Suratose T. et al [8] proposed the two-factor authentication protocol for USB-type data storage devices. This scheme uses username and password together with HMAC-based one time password (HOTP). The researchers aim to add security

level of the devices so an unauthorized user will not allowed to use the device or make a copy version and distribute to other devices and run the application. Only the authorized users are able to bind the specific USB data storage device with the secret code one-by-one. The code-embedded USB device can use for a limited period. After that it will be useless and needs to be returned to an authorized user for re-initialization. The purposed authentication protocol consists of 5 steps during initial setup process. By using Diffie-Hellman [9] key exchange parameters, one-way hash function, and asymmetric key exchange, these methods has ability to encrypt the messages during the secured connection establishment. The researchers claimed that the purposed protocol can prevent the messages from many attacks. The strong point of this scheme is that only the legitimated user with smart phone can uses the specific USB data storage device for the assigned job. Even if the legitimated user wants to distribute the data, but the data cannot be distributed to the other device because the software had bind the code to the unique parameter of the USB device,

The next research will focus on One-time Password authentication scheme purposed by Jiang et al. [10]. This research purposed a simple and effective scheme based-on SM2 algorithm and hash function. Using two pair of asymmetric encryption keys and hash function to create strong authentication scheme without needs of symmetric encryption keys. This scheme was separated into 2 main phases. In the registration phase, the client will send his identities information through secure channel to the server. Then the server will generate some values and store for further uses. During the authentication phase, client and server will exchange using only public keys of data encryption.

Lastly but not least, the paper studied the One-time password-based two-factor authentication scheme purposed by Liu et al [11]. The purposed scheme used two different hash functions during the process, one for generating the core hash chain whereas another one is for plain-text encryption during transmission. The scheme is also separated into 2 phases, the registration phase and the authentication phase. In the registration phase, user can get two hash functions, which are different from each other, for using with the secret key *seed* to generate the hash chain. Then both user and server will discard the seed. In the authentication phase, after user accesses to server and enters his username and password, the core hash chain will send as challenge code

to the server. The server then creates and sends the challenge answer back to the user. The user compares the parameter, if it is correct and the server is trusted. Then the user generates OTP and sends to compare with another calculation by server. If both OTP is matched, then the server will provide it servers to the user.

However, most of reviewed is this subsection did not focus on using in the Digital Forensic field, so we have to design and implement the authentication protocol which is appropriated to use in the Digital Forensic applications.

2.2 Verification using Formal Method Tool

To ensure that the purposed security protocol achieves the goal, the protocol should be deeply examined into each running state. If the finite state of the protocol is satisfy the goal, the protocol is secure. The verification is required to investigate the completeness and correctness of the purposed protocol.

Coloured Petri Nets (CPNs or CP-Nets) is a graphical-based mathematic method that is able to design and construct models, including simulation and verify the system which major in communication, synchronization, and sharing resources. [12, 13, 14]. CPNs were developed from Petri-Net which has strength in presenting the concurrent system together with key feature of programming, which is defining and managing the data [15]. These make the model constructed by CPNs, they are also able to describe the state in the system or changes between the states.

As shown in Figure 2.1, the CPNs workspace consists of three things, the circles and ellipses are the places which describe the state in the system. The rectangles are the transitions which describe the actions. The arrows are the arcs which present how the state changes when transition occurs. Arcs only connect between place and transition, not between only places or between only transitions. The place which has an arc point out to the transition is the input place. On the other hand, the place which has an arc point in from transition is the output place of that transition. Each place consists of tokens and each token contains the data which are corresponding to their types, the tokens are called color. These make CPNs different from an old Petri-Net which is a low-level language.

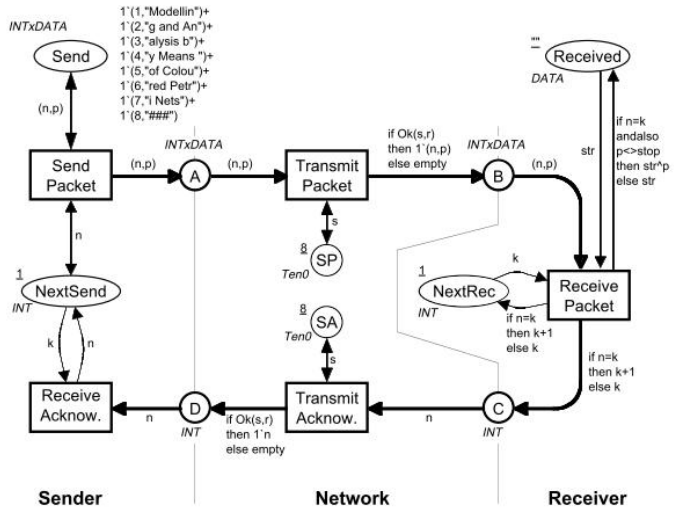

Figure 2.1 An example protocol model

The first research of this subsection is the thesis of Ph.D. that was introduced by Tritilanunt S. [16]. The thesis was introduced the protection on protocol against Denial-of-Service attack (DoS) which is used the CPNs tools for modeling and analyzing the protocol. The advantages which CPNs had over other models and analysis tools are interactive simulation, time concept, and large number of formal analysis methods which can be proved. The most important advantages of using Coloured Petri Nets are not only the graphical presentation and dynamic behavior for visualize systems, but it also prevents the inconsistency of the message sequences among the protocol participants in the design of the protocols. In particular, Coloured Petri Nets also have many advantages over original Petri-nets. Since CPNs supports a hierarchical modeling concept, CPNs is able to handle large network with a number of data set. Moreover, the CPNs have a feature to reduce the complex model size by holding many tokens in similar set within a single place. For these reasons, we use CPNs tools in this research for modeling and analyzing the authentication protocol in order to verify the possible attacks which many occurrences during the establishment of the protocol.

For the research proposed by Basyouni and Tavares [17], this paper introduces cryptographic protocol automated analyzing method using Coloured Petri-Net. The method is based on backward state analysis [18] and the reachability matrix description for Coloured Petri-Net [19] which consists of two steps. The first step is

Acceptance Check Step (ACS) which use to find the set of modified output data and to identify the vulnerable points in the protocol. The second step is the Matrix Analysis Step (MAS), which gives the result that can indentify whether the insecure state is reachable. The researcher also applied this method to the security protocol which presented by Aziz and Diffie [20] for analyzing. The result proved that the analyzed security protocol have got some vulnerable states and insecure state which can be reachable from the initial state.

The lastly paper is about the method to model and analyze the security protocols proposed by Xu and Xie [21]. However, the researchers found the problem on the tradition CP-Nets intruder model which is applicable only one attack and can describe one insecure state. To solve the problem, the researchers had proposed the new finite state analysis method using CPNs. By modeling the protocol as a CP-Net, CPNs tools are used to automate protocol analyzing. Not only purposing the new analysis method, but the researchers also presented a new intruder model. In order to solve the problem in the tradition intruder model, the researchers considered about the abilities of the intruder, participated in the public channel, which consists of many types of attack. By the assumption from Dolev-Yao model [22] ,the intruder model is adopted to overcome the problem. The new model can be considered as two concurrent processes. The first is waiting for the message interception; whereas the second process performs the selected attack depending on the protocol sessions. The new model constitutes of many types of possible attack on protocol which the intruder can combine the attacks depending on the messages to intercept the communication. The final contribution, the paper gives an example of using the new method on the Andrew secure RPC Protocol. After model checking and state space analyzing, the attack was found as same as Gavin Lowe [23] discovered.

This thesis is going to use the finite state analysis method to analyze the proposed protocol. The unsafe state is created to search for the possible attack in the model of the protocol. If there is a token in the unsafe state at the final state, the analysis process can detect that there is the possible attack in the protocol.

From reasons above, the CPNs tools was able to use for implementing and verifying the authentication protocol. So we choose the CPNs tools and a verification tool for the proposed authentication protocol.

CHAPTER III
RESEARCH METHODOLOGY

This chapter describes the development process of the proposed protocol to add the security layer in USB data storage devices. Our protocol is an authentication protocol which is based on two-factor (or two-step) authentication. The *username* and *password* are the first factor in the authentication process, followed by *one-time password* (or OTP) as the second factor. The OTP is generated from many unique factors in order to limit the usage for specific person and devices. The research is divided into six phases, including the Studying phase, the Design phase, the Verification phase, the Implementation phase, the Validation phase and the Analysis phase as shown in Figure 3.1

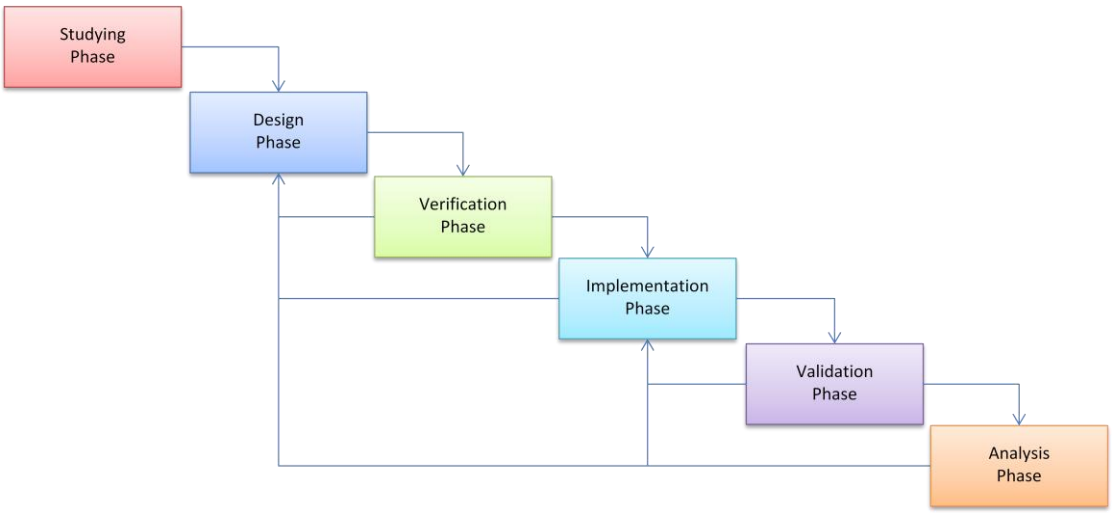

Figure 3.1 Research Methodology

3.1 Studying phase

The purpose of this phase is to study the research and review the literatures related to designing the authentication protocol and the formal method tool such as Coloured Petri-Nets, as shown in Chapter II. Then we compare among the several authentication protocols in order to develop the appropriated authentication protocol for digital evidence acquisition device.

3.2 Design phase

In this phase, we design the proposed authentication protocol for secure the usage of USB data storage device by using two-factor authentication scheme. Because the proposed protocol is designed for apply to digital evidence acquisition devices in digital forensic applications. The proposed protocol must be able to prevent unauthorized access, limit the access time, and the duplication of software from the device to others. We also design the protocol with the light-weight communication properties between participants, by reducing the number of transmission messages for fast connection while maintaining the security level of transmitted messages.

All abbreviation and acronyms of the purposed protocol will be described as the followings.

3.2.1 Abbreviations and Acronyms

The followings are the parameters and symbols used in this research:

A. Parties

- Sv : Server or Administrator
- C_X : User X

B. Messages

- ID_X : The user U_x's identity
- PW_X : The user U_x's password
- USB_X : Unique serial number, PID (product identifier) and GUID (globally unique identifier) of USB storage device X
- IMEI : Unique serial number of Smartphone

- OTP_X : One-time password which is generated only for user X and USB device X
- $E_{Ks}(\bullet)$: Symmetrical encryption using session key Ks on (•)
- $D_{Ks}(\bullet)$: Symmetrical decryption using session key Ks on (•)
- H(•): Message Digest of Hash function apply on (•)
- N : Random nonce generated by server
- T : Timestamp

3.2.2 Protocol Overview

This subsection presents the model of the proposed protocol as shown in Figure 3.2, and describes each message which is transmitted between the participants.

1. $C_A \rightarrow Sv: g^x \bmod p, ID_A, H(Pass), USB_A, IMEI$

First, the client A generates the Diffie-Hellman key exchange parameter g^x under modulation of *p*, where *g* is a primitive root under large prime number *p*, and *x* is a number which is randomly chosen under (*p-1*). The client *A* attaches $g^x \bmod p$ with the other parameters which consist of user *A*'s identification (ID_A), password in hash

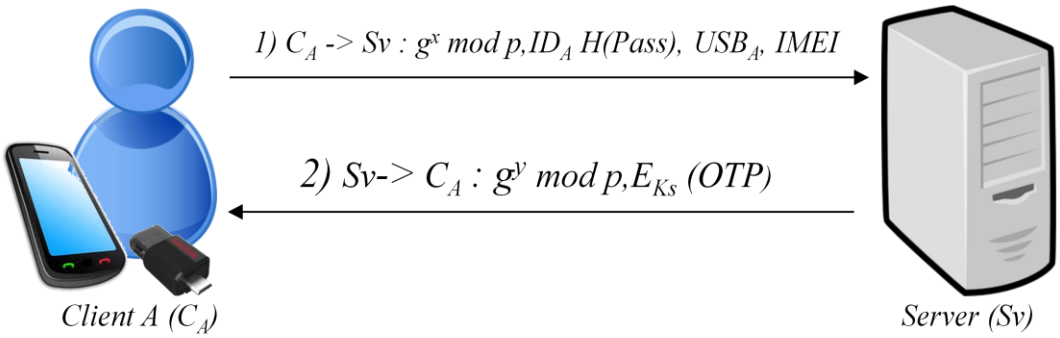

Figure 3.2 Protocol Overview

format (*H(Pass)*), unique number of USB device of client A (USB_A), and smartphone unique number (*IMEI* : International Mobile Station Equipment Identity). After message creation is finished, the client *A* sends a requested message to the server.

When the server receives and separates the message into two parts; session key parameters and *OTP* generation parameters. Before responding to the client, the

server needs to authenticate client by checking the identity and hash password of the client *A*. If the value is valid, the server continues to create message in the next step. Otherwise, server rejects the packet and closes the session.

2. Sv → C_A: g^y mod P, $E_{Ks}(OTP_A)$

Server generates g^y with the conditions as same as the g^x. Then the server is able to generate the symmetrical session key K_s. While the second part is used for generate the One-time password (or *OTP*) for the USB data storage device, which is owned by client *A* (USB_A) according to the unique number of the device. The purpose of using user's parameter and unique parameter of USB bundled with the IMEI code of smartphone is that we can limit the usage of USB device for assigned user and task only. Moreover, the gathering tool stored in the USB device cannot be distributed to other devices, because the OTP generated by using our technique cannot be used by an unauthorized user using an USB device which has different serial number.

By responding to the client, the server encrypts the OTP_A by using the session key K_s and attaches the encryption with the generated g^y *mod p* before sending the message back to the client *A*.

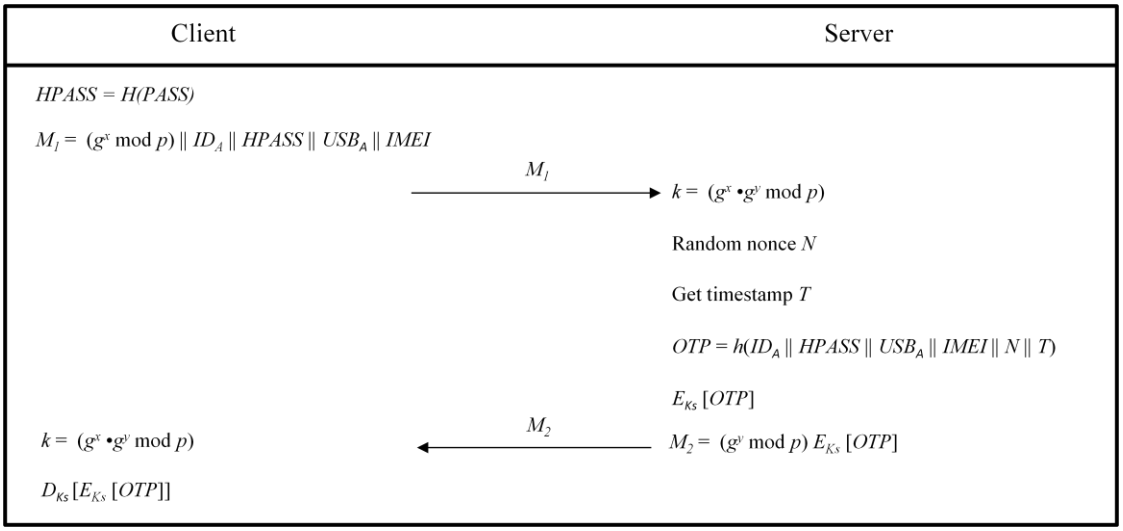

Figure 3.3 Authentication Phase of the proposed protocol

3. $C_A \leftrightarrow Sv$: E_{Ks}*(message)*

When the client *A* receives the message, he separates it into two parts; the $g^y \ mod \ p$ part and the encryption of OTP_A. Before the client *A* is able to decrypts the encryption part in order to get the *OTP*, the client *A* needs to compute the session key K_s by calculating the Diffie-Hellman algorithm. Once the client *A* obtains OTP_A by decrypting the second part, the client can start using USB device to collect data evidence, as well as to access the data stored in that USB device.

Optionally, if the client *A* and the server want to further communicate, both principals are able to transmit the messages securely by using this symmetric session key K_s.

3.2.3 Creating the proposed protocol in CPN Model

After we finish designing the protocol model as shown in Figure 3.2. Next, we create the protocol model using CPN tool in order to ensure the security and correctness of workflow in the protocol.

3.2.3.1 Declarations

Similar to other programming languages, the CP-Net also needs to declare all the variables which were used in simulation the model. There are two category of variable in the CPN Tools; Simple Color sets and Compound Color sets. In this proposed model, we uses two type of simple color set and two types of compound color set as the following;

Integer color set The basic color set which consists of numeral without decimal point. This color set type is used to contain the numeral values.

String color set Another basic color set which used to contain the string of ASCII characters. This color set type is used to store the message token.

Record color set One of the compound color set which has a fixed-length values. Each of the component color sets may be a different type. In this model, the record color set will be used as an encrypted message.

Enumeration color set This is another compound color set which is used in the model. The enumerate values are explicitly named as identifiers in the declaration.

In this model, the enumeration color set will be used in the condition that may need to contain difference value token.

Figure 3.4 Declarations of the proposed protocol in CPN tools

Refer to Figure 3.4, two color sets; *INT* and *STRING*, are declared to the set of integers and strings accordingly. The next nine unit color sets are *ID* for user identification token, *PASS* for password token, *KEY* for Diffie-Hellman key exchange parameters *gx*, *gy*, and *gz*, *SKEY* is the symmetric session key generated from Diffie-Hellman key exchange parameters for the communication in the protocol *K_AB*, *K_AZ*, and *K_BZ*, *USB* for unique number of USB data storage devices belong to client A (*USB_A*) and Intruder (*USB_Z*), *SEEDSV* for initial seed token *SEED_SV* in the server, *IMEI* for smartphone's *IMEI* token, *OTP* is the One-Time passwords which were generated for client A (*OTP_A*) and intruder (*OTP_Z*), and *MODE* for scenario selection *E1*, *E2*, and *E3*. The four record color sets, *M1*, *M2*, *REQ*, and *RES*, are declared to the messages which are transmitted in the protocol. The record color sets, *MX1*, *MY1*, *REQ1*, and *RES1*, are declared to the part of messages which are separated by the principles. We declare variables *id* and *id_n* of type *ID*, *key* and *key1*

of type *KEY*, *skey* and *skey1* of type *SKEY*, *pass* of type *PASS*, *usb* and *usb1* of type *USB*, *imei* of type *IMEI*, *mode* of type *MODE*, *m1* of type *M1*, *m2* of type *M2*, *mx1* of type *MX1*, *my1* of type *MY1*, *seedsv* of type *SEEDSV*, *otp* of type *OTP*, *req*, *reqt*, *res* and *rest* of type *REQ*, *REQT*, *RES* and *REST* accordingly, and *n*, *p* of type *INT*.

3.2.3.2 Coloured Petri-Nets Models

To verify workflow of the proposed protocol, we construct the model for protocol validation and verification the workflow. The model participants are the client, the server and the intruder.

Protocol model overview

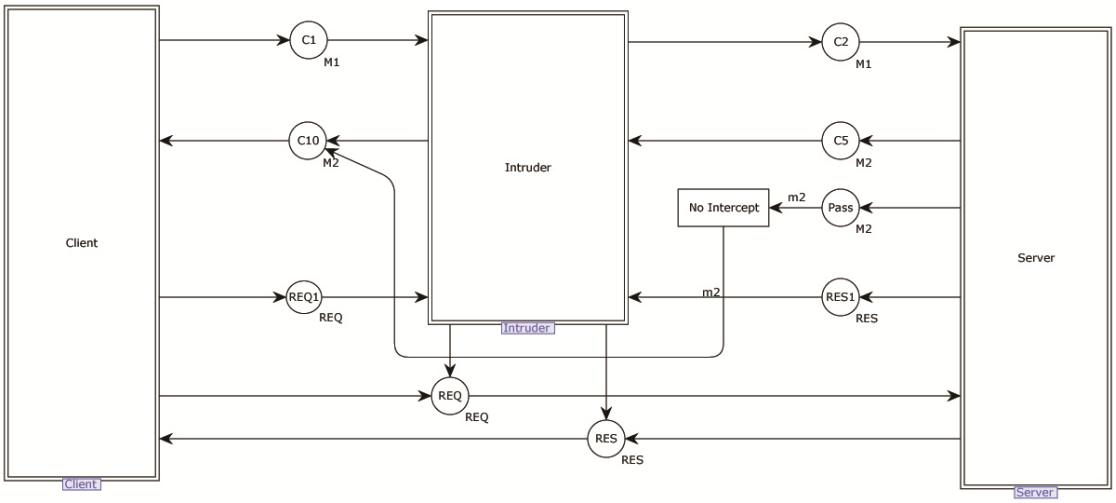

Figure 3.5 Protocol overview in CPN tools

Figure 3.5 shows the overview of the proposed protocol and how communication messages flows. From the figure, the client begins by sending the message *M1* to the server pass through the intruder. By this means the intruder can choose to intercept the communication message *M1* or not. The server receives message *M1*, then the server responses with message *M2*. At this point, having chosen to intercept the message *M1*, the intruder has to intercept message *M2* too. On the other hand, if the intruder had left the message *M1* and passed to the server without intercept, the message *M2* will transmit pass through *"No Intercept"* transition and direct to the client. For the further request, the client sends the message *REQ* to the server through place *REQ*, if there was no intercept from the intruder at the beginning.

But if the intruder had initiated the interception the communication, then the request message *REQ* will pass to the intruder through place *REQ1* before go to the place *REQ* and to the server accordingly. The response message *RES* from the server is as same as the request message *REQ*. The concepts of this workflow will be described later in this chapter.

After review the overview and work process of the proposed protocol, then we will analyze each participant's protocol deeply. In order to understand how the participants run the communication of the protocol.

Client side model

The model of the client consists of many places, each of them contains tokens of unique parameters including an identification of client A (*ID*), a password in hash-form (*PASS*), unique number of USB device owned by client A (*USB_A*), smartphone unique number (*IMEI*), and Diffie-Hellman key exchange parameter (*gx*). The protocol starts with the "*Craft M1*" transition binds these tokens into a message *M1*coloured token and sends it to the server. Meanwhile, some tokens which are *ID* and *gx* are brought back to their place for further use.

Note that the "*Mode*" place which contains tokens *E1*, *E2,* and *E3* is the protocol scenario selector for different simulation setting in the model. This place will be explained how it works, later in the intruder model topic.

After receiving the *M2* coloured tokens from outside (shown as place *"C10"*), the message is separated into two tokens; the Diffie-Hellman key exchange parameter token and the encryption of OTP coloured token. The former token is sent to combine with *gx* token and generate the symmetric encryption key token then stored in the place *A4*. The second coloured token brings the key token from where it placed to decrypts the encryption of OTP coloured token, then sends the key token back to place *A4* after finished the decryption. The client will use the OTP token which is retrieved from the decryption, if the OTP token is a valid *OTP_A* then the protocol is secured. If the OTP token is not a valid *OTP_A*, then the client cannot use this OTP which means that there is an intruder in the protocol. The model then generates the *insecure* token and moves it to the "*Unsafe*" place (shown in red) to represent that the protocol is unsafe.

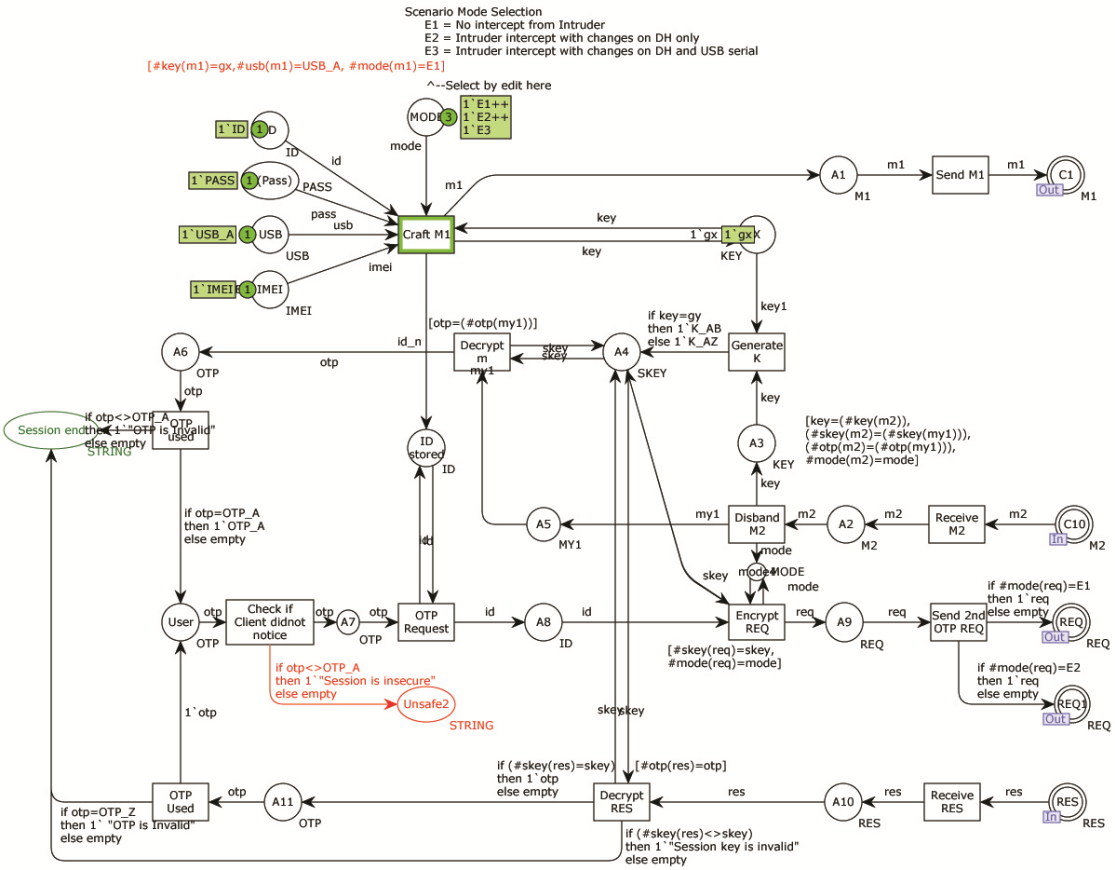

Figure 3.6 Client model in CPN tools

For further communication, client uses *ID* token which is encrypted by the generated session key token to create the *REQ* coloured token for requesting a new OTP and sends to the server via place *REQ* or *REQ1*. When receiving the new OTP response message coloured token (*RES*) from outside, the session key token is needed to be verified by comparing with the previous session key. If the result is mismatched, then the protocol will be suspended. After session key verification, the new OTP token will be used. If the new OTP is not *OTP_A* for *USB_A*, then the protocol will also terminated as well.

Server side model

After receiving the message *M1* coloured token from outside (shown as place *C2*), the server then separate the *M1* into three tokens. The first one is *ID* token which will be kept in the database for further verification. The second one is a *KEY* token which is brought to combine with *gy* key token and generate the session key *SKEY* token. The last one is a coloured token which contains a set of parameters

consist: *id*, *pass*, *usb*, and *imei* tokens. This coloured token is combined with the initial seed *SEED_SV* token in the server to calculate the *OTP* token. The server encrypts the *OTP* token with *SKEY* token then attaches with *gy* key token to create the message coloured token *M2* and returns *M2* to a client. At this point, there are two paths for message *M2* to be taken which is depend on value of mode token. This will be described later in this chapter.

The new *OTP* token is generated by using *SEED_SV* and previous *OTP* token. The counter token will be moved back to the "*Counter1*" place and a tool will increase the number which is used to limit the usage number of USB device. When the new OTP token is generated, it will be encrypted with *SKEY* and sent back as a response message *RES* coloured token.

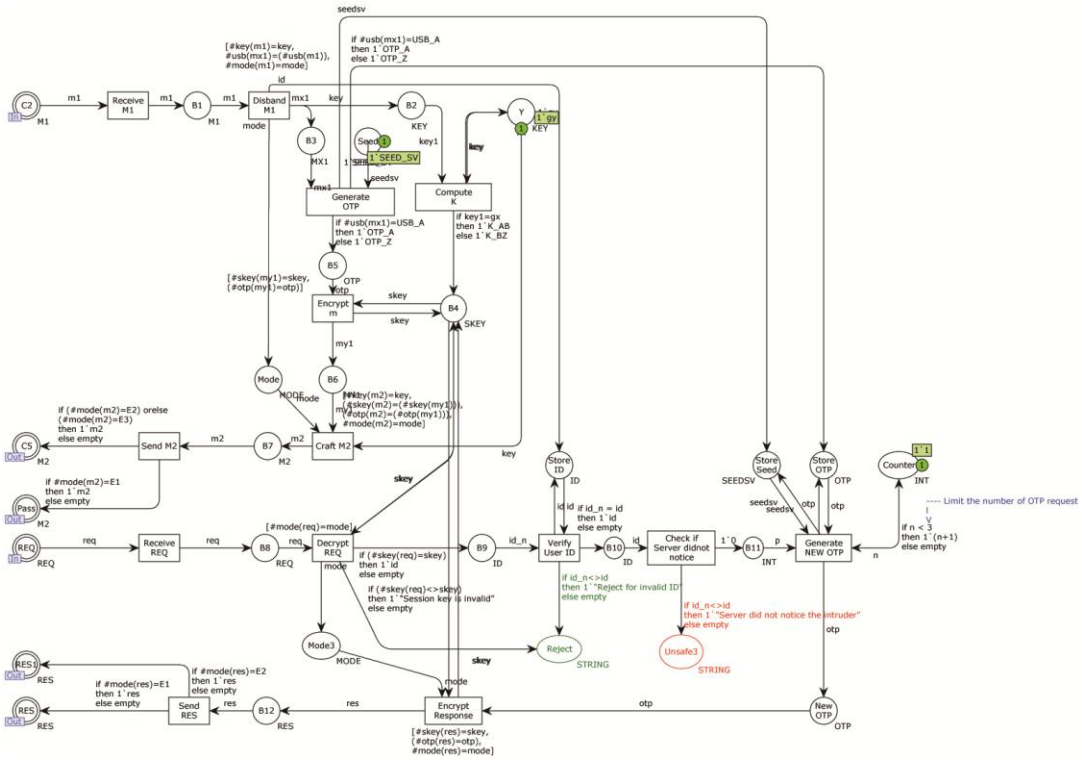

Figure 3.7 Server model in CPN tools

Intruder side model

The intruder model is placed between the client and the server communication channel and taps the channel for sniffing the messages and/or alters them. The way that the intruder taps and alters the message is called Man-in-the-Middle attack, which is one of the well-known attacks in the network communication.

To achieve the goal of the protocol, we create the intruder model in the protocol and give the assumption of the intruder's ability that is able to carry out the following actions:

1. Intruder can intercept the transmission without noticed by both client and server

2. Intruder can modify Diffie-Hellman key exchange parameter and unique number of the USB device in the message during the transmission

3. Intruder can run Man-in-the-Middle attack and modify a pair of session keys between client and-intruder and intruder-server.

3.2.3.3 Scenarios

To ensure that the proposed model can satisfy the goals, we create three scenarios that represent difference attacking simulation from the intruder in the same model. We also create *Mode Selector* place in order to select the scenario to run, and control the workflow of each simulation in the model as shown in Figure 3.9.

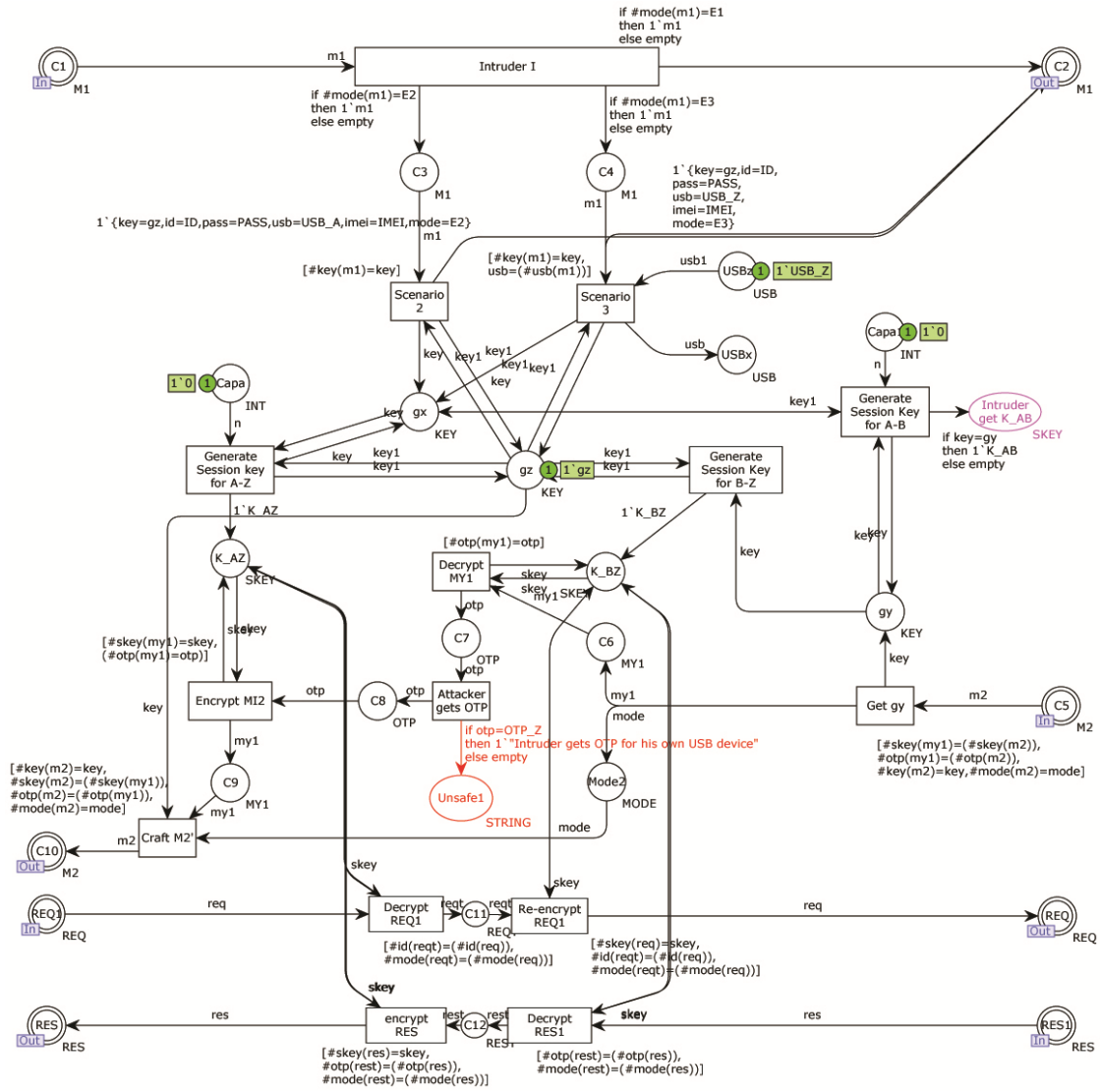

Figure 3.8 Intruder model in CPN tools

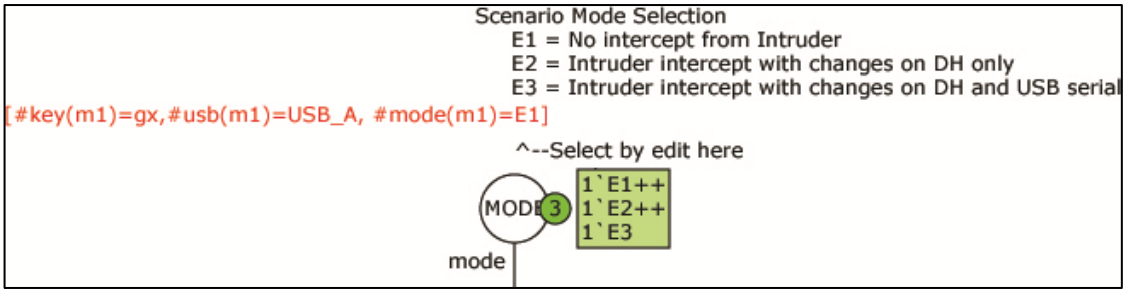

Figure 3.9 Mode Selector in CPN tools

The Mode Selector place contains three token at the beginning state, *E1*, *E2* and *E3* token, which needs to be assigned before starts the simulation, by editing

the value of *#mode(m1)* at the end of the condition string (shown in red). The simulation conditions of each mode are also described above the condition string.

Mode E1: The model are not being attacked

The protocol starts with the client creates *M1* coloured token by binding the client A token(*ID*), a password in hash-form token (*PASS*), unique number of USB device, which is owned by client A token(*USB_A*), smartphone unique number token(*IMEI*), Diffie-Hellman key exchange parameter token(*gx*) and Mode Selector token (*E1*, *E2* or *E3*). Then the client sends the coloured token *M1* out through place *C1* as shown in Figure 3.11.

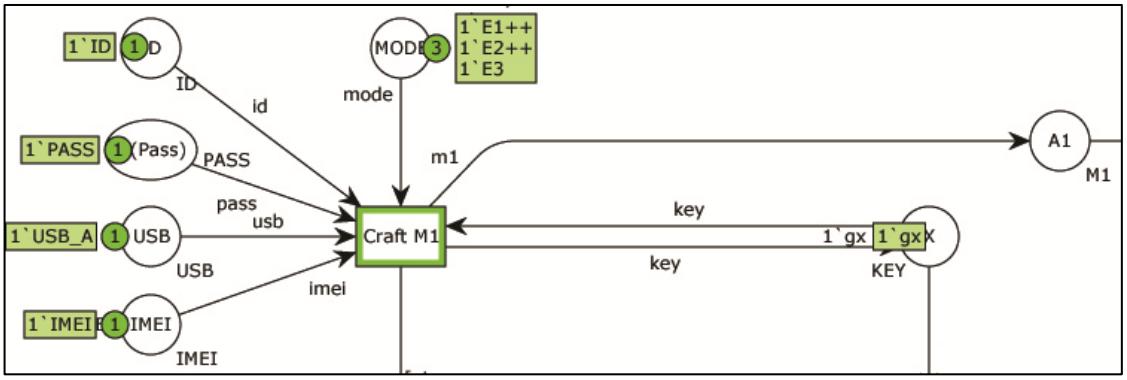

Figure 3.10 Mode *E1* at the beginning state

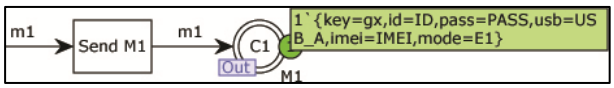

Figure 3.11 Message *M1* coloured token before sends out

The intruder receives the message *M1* coloured token from the client, then forwards the message *M1* to the server without changing the message due to mode *E1* does not has any attack from the intruder (as shown in Figure 3.12).

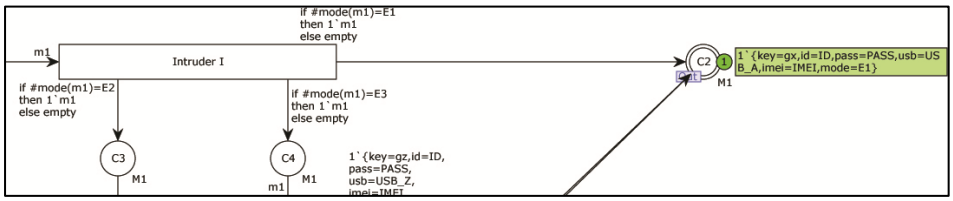

Figure 3.12 Intruder forward message *M1* to the server without change

After the server receives the originally message *M1* from the intruder, the server separates the message *M1* coloured token into three part (exclude the mode *E1* token). The first one is *ID* token which is stored for further use. The second one is Diffie-Hellman parameter *gx* token which is brought to combine with *gy* key token and generate the session key *K_AB* token. The last one is a coloured token which contains a set of parameters consist: *ID*, *PASS*, *USB_A*, and *IMEI* tokens. This coloured token is binds with the initial seed *SEED_SV* token which is generated by the server to calculate the *OTP_A* token for client as shown in Figure 3.13. The server also stores *OTP_A* token for further use as same as the *ID* token.

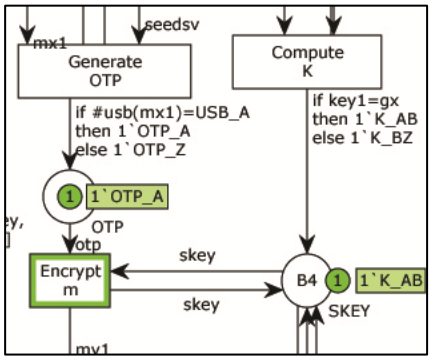

Figure 3.13 Result from OTP generation and encryption key

The server then encrypts the *OTP_A* token with generated key *K_AB* token and binds with *gy* token before send back to the client. Because there is no intercept from the intruder (mode *E1*), the message *M2* coloured token is sent through place *PASS* (shown in Figure 3.14).

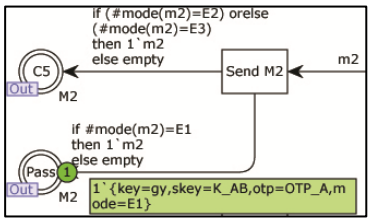

Figure 3.14 Server sends message *M2* though place *Pass*

When the client receives the message *M2* coloured token, the message is separated into two parts (exclude the mode *E1* token). The first part is Diffle-Hellman key parameter *gy* token from the server, which is sent along with *gx* token from the client himself to generate the symmetric encryption key *K_AB* token. The second part is the encryption of *OTP_A* which will be decrypted by using the generated *K_AB*. After decrypting the encryption of *OTP_A*, the client gets *OTP_A* token for use with his belonging USB data storage device.

For further communication or client requests the new OTP for the next access permission, the client has to create the request message *REQ* coloured token by using encryption key to encrypt the user *ID* token and sends to the server. Because this is in mode *E1* which is no attack from the intruder, the message *REQ* coloured token is sent through place *REQ*.

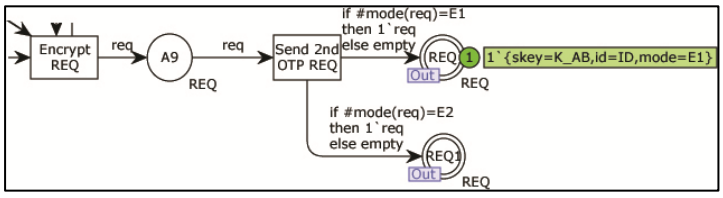

Figure 3.15 Client sends message *REQ* though place *REQ*

The Server receives message *REQ* coloured token and decrypts by using encryption key *K_AB* to get client *ID* token. The decrypted *ID* token will be compared with the previous stored *ID* token to verify the user identification. If the *ID* token is valid, then the server generates the new *OTP_A* token for the request. Then, the server binds the *SEED_SV* token and previous *OTP_A* token to create the new *OTP_A* token. This process is controlled by place "*Counter*", from Figure 3.16. The condition from the transition to place "*Counter*" is "*if n<3, then 1` (n+1), else empty*" (indicated with

blue message) which means the client can request the new OTP from server only three times. In the model, we can change this limitation by editing this value in the condition to a favorite number.

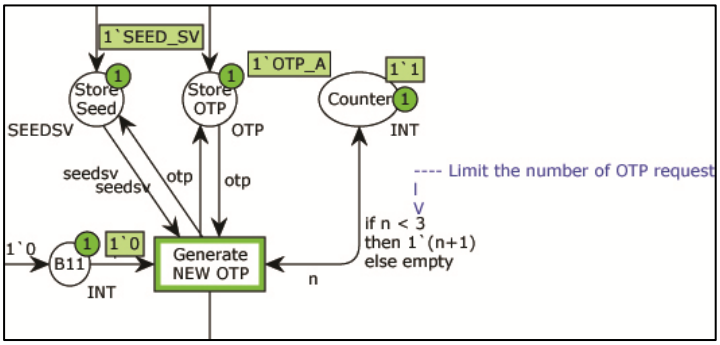

Figure 3.16 Server generates the new *OTP_A* token

After generating the new *OTP_A* token, the server encrypts the token using encryption key *K_AB* token to create response message *RES* coloured token before sending back to the client. Since there is no attack from the intruder in this scenario, the message *RES* coloured token is sent through the place *RES* following the mode *E1* condition (Figure 3.17).

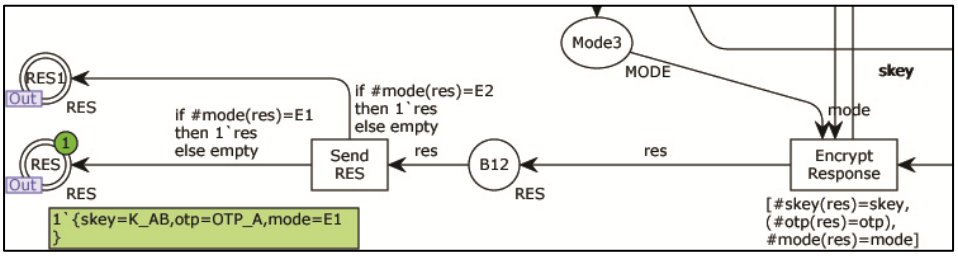

Figure 3.17 Server sends *RES* coloured token

The client receives the message *RES* coloured token from server and decrypts the message to get *OTP_A* token by using key *K_AB* token. Then the client can use the new OTP for an assigned task. The client can also request the new OTP again until the place *Counter*'s condition in the server is reached the limit. If the condition is reached, then the client cannot request the new OTP anymore (Figure 3.18).

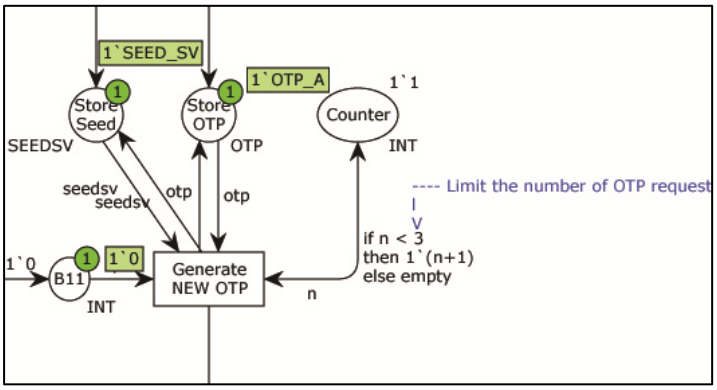

Figure 3.18 Server cannot generate new OTP

This scenario is a normal workflow of the proposed protocol, so we can investigate the running process step-by-step to verify the correctness of the protocol in normal situation.

<u>Mode E2</u>: The intruder attacks the protocol using Man-in-the-Middle attack and changes Diffie-Hellman key exchange parameters

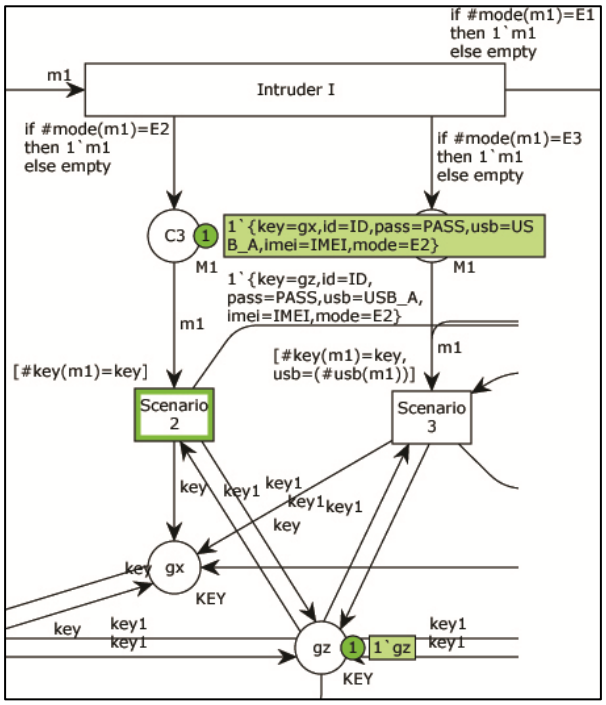

Figure 3.19 Intruder intercepts the message *M1*

This scenario starts as same as the Mode *E1* scenario, the client creates message *M1* coloured token and send out to the server through the intruder. But this time, the intruder intercepts the message *M1* and reroute the message to place *C3* as shown in Figure 3.19. The intruder initiate the Man-in-the-Middle attack, by replace Diffie-Hellman key exchange parameter *gx* with *gz*. Then the intruder forwards the modified message *M1* coloured token to the server while the *gx* token is stored in place *gx* for further use.

The server receives the message *M1* coloured token and separates the message into three parts as same as the previous scenario. But the result of encryption key generation is *K_BZ* instead of *K_AB* token, because the pair of Diffie-Hellman key exchange parameter tokens *gz* and *gy* instead of *gx* and *gy* (Figure 3.20).

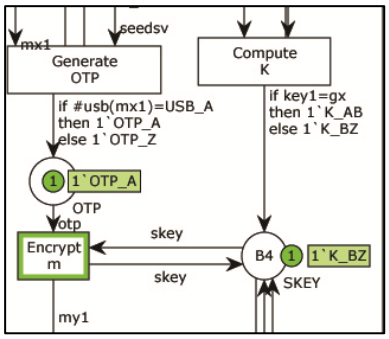

Figure 3.20 Encryption generation result

The rest of the process which happens in the server is as same as the previous scenario in mode *E1*. The difference is when the server is going to send the message M2 coloured token out, the message *M2* transmits through the place *C5* instead of place *PASS* as shown in Figure 3.21. This is because the intruder intercepted the communication and changed the Diffie-Hellman key exchange parameter tokens. If the intruder lets this message pass to the client without intercept, then the client will notice the attack, because the client cannot decrypt the encryption of the message.

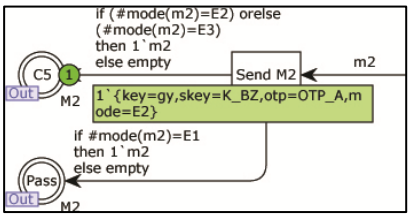

Figure 3.21 Server sends the message *M2* via place *C5*

The intruder receives the message *M2* and separates into two parts; the Diffie-Hellman key exchange parameter *gy* part and the *OTP_A* token encryption part. The first part binds with the stored *gx* token to create the symmetric encryption key *K_AB* token, and *gz* token to create the *K_BZ* token. The intruder uses the *K_BZ* token to decrypts the encryption of OTP to retrieve the *OTP_A* token as shown in Figure 3.22.

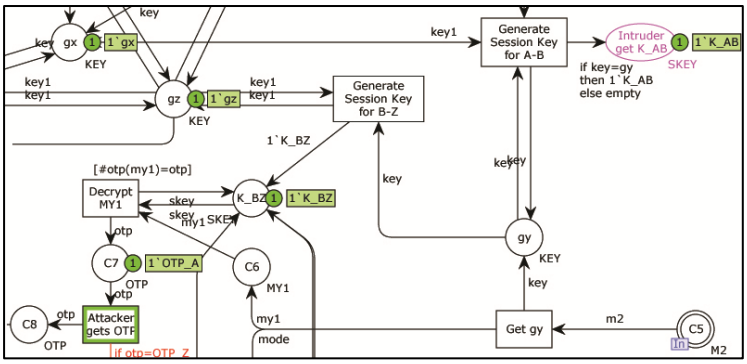

Figure 3.22 Intruder gets *OTP_A* token

The intruder also binds the Diffie-Hellman key exchange parameter *gx* and *gy* to create the encryption key *K_AZ* token. Then the intruder uses *K_AZ* token to encrypt the *OTP_A* token and forwards to the client through the place *C10*. By doing this, the intruder can avoid the client noticing the Man-in-the-Middle attack from the intruder. After receiving the modified message *M2* coloured token, the client separates and decrypts the message as same as the process in mode *E1*. This time the client uses the received *K_AZ* token to retrieves the *OTP_A* token (Fig 3.23).

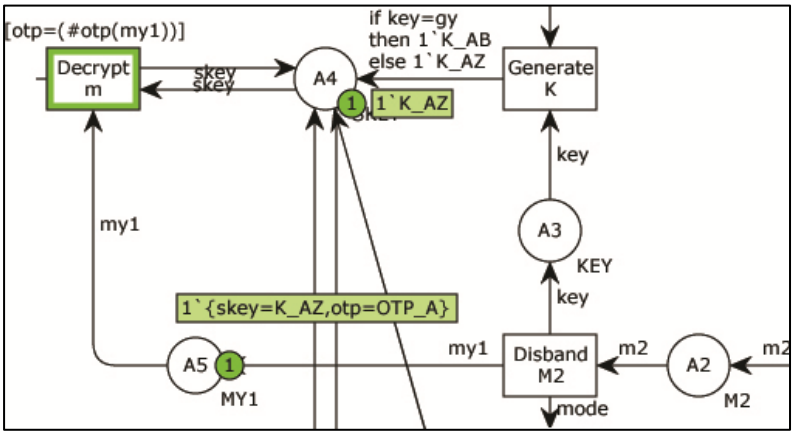

Figure 3.23 Client gets *OTP_A* token

For addition *OTP* requests, the client can also create the request message *REQ* coloured token and send to the server. In this scenario, the request message *REQ* is sent through the place *REQ1* instead of *REQ* due to the value of mode selector token is *E2* as shown in Figure 3.24.

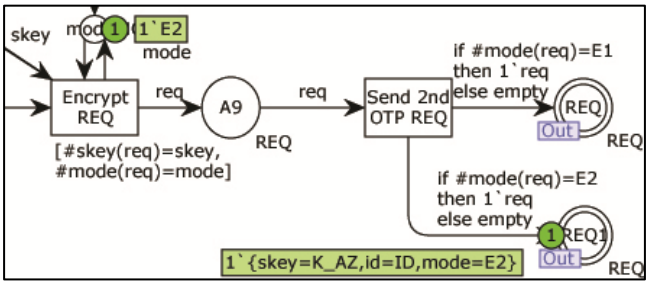

Figure 3.24 Client sends request message *REQ*

The intruder intercepts the request message *REQ* then decrypts with *K_AZ* token and re-encrypts with *K_BZ* token, then forwards the request message to the server (Figure 3.25). The reason is if the intruder leaves the request message *REQ* pass without replacing the encryption key, the server is unable to decrypt the encryption with *K_BZ* token. Then the attack will be compromised and the session will be terminated.

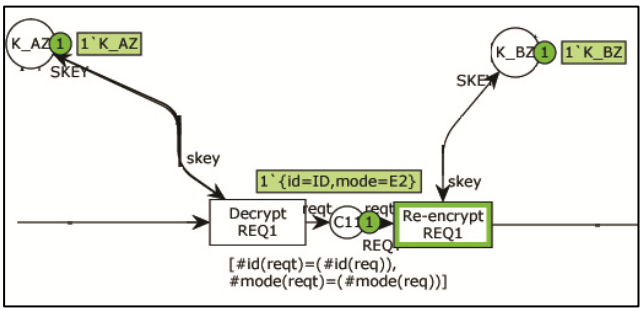

Figure 3.25 Intruder intercept and replace the encryption of the message

The server receives the request message *M2* coloured token from the intruder, the server decrypts the message and retrieves the *ID* token. The process and conditions of generating the new OTP is as same as in mode *E1* simulation. After generating the new *OTP_A* token, the server decrypts *OTP_A* by key *K_BZ* and send back to the client through the place *RES1* (Figure 3.26).

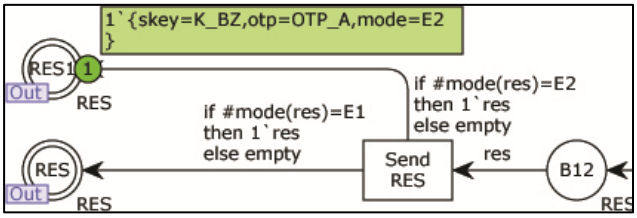

Figure 3.26 Server sends response message *RES*

The intruder also intercepts the response message *RES* from the server, by decrypting with *K_BZ* and re-encrypting with *K_AZ*, in order to avoid the client notices the attack.

<u>Mode E3</u>: The intruder attacks the protocol using Man-in-the-Middle attack and change Diffie-Hellman key exchange parameters and USB unique serial number

This scenario also starts as same as the Mode *E1* scenario, the client creates message *M1* coloured token and sends out to the server through the intruder. But this time, the intruder intercepts the message *M1* and reroutes the message to place *C4* as shown in Figure 3.27. The intruder initiates the Man-in-the-Middle attack by replacing not only Diffie-Hellman key exchange parameter *gx* with *gz* but also

replacing USB unique serial number *USB_A* with *USB_Z*, then forwards the modified message *M1* coloured token to the server while the *gx* token is stored in place *gx* for further use.

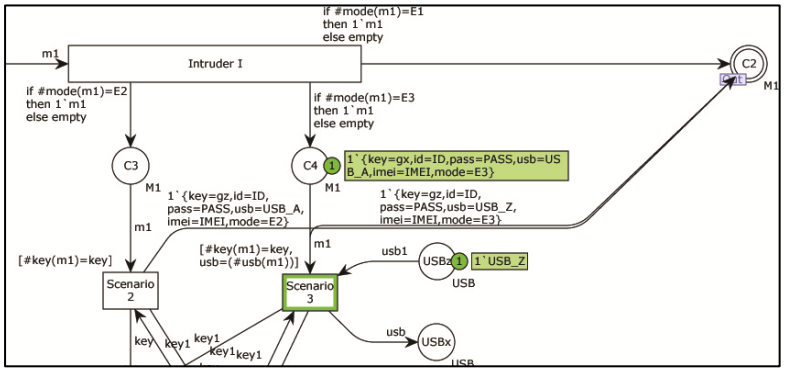

Figure 3.27 Intruder intercepts the message *M1*

The server receives the message *M1*, separates the message and generates the encryption key *K_BZ* as same as the previous mode *E2*. But the OTP generation is different, the server uses *OTP_Z* token instead of *OTP_A* token to generate the OTP. Figure 3.28 shows the result of the generation is *OTP_Z* token.

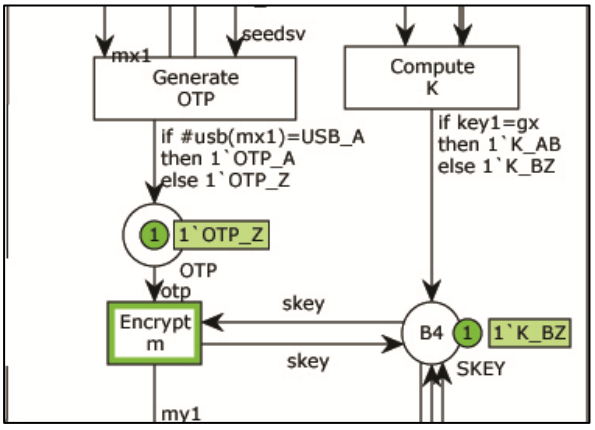

Figure 3.28 *OTP_Z* token is generated

The rest of the message *M2* generation process is as same as the previous scenarios (mode *E1* and *E2*). The message *M2* coloured token consists of Diffie-Hellman key exchange parameter *gy* token and *OTP_Z* token, which is encrypted by

K_BZ token. The message M2 is sent through the place *C5* because the value of mode selector token is *E3*.

The intruder receives the message *M2* and separates the message into parts. The *gy* token is sent to bind with token *gx* and *gz*, to generate the key token *K_AB* and *K_BZ* accordingly. The key token *K_BZ* is used to decrypt the encryption of *OTP_Z* token in order to retrieves the *OTP_Z* token. Because the intruder gets the *OTP_Z* token, the place "*Unsafe1*" has a token "*Intruder gets OTP for his own USB device*" (Figure 3.29).

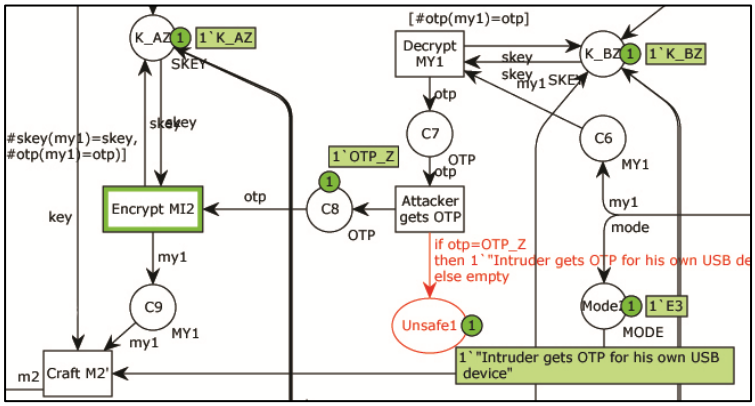

Figure 3.29 Intruder gets the *OTP_Z*

The intruder continues to encrypt the *OTP_Z* token and binds with *gz* token, before sends out to client. The client receives the modified message *M2* coloured token and splits into parts. The Diffie-Hellman key exchange parameters *gz* is taken along with *gx* to create the encryption key token *K_AZ*, which is used to decrypts the encryption of *OTP_Z* as shown in Figure 3.30.

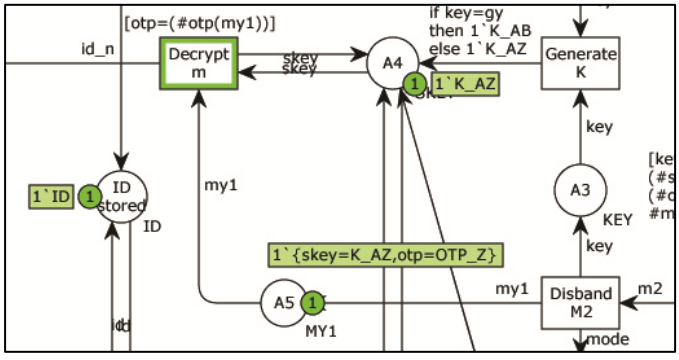

Figure 3.30 Client decrypts the OTP encryption

The client retrieves the *OTP_Z* token from the encryption, and notices that the retrieved OTP is not a *OTP_A* token. Finally, the place "*Session End*" has the token "*OTP is invalid*" stored in the place (Figure 3.31).

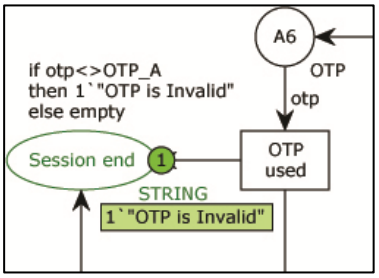

Figure 3.31 Client notices that OTP is invalid

The last scenario we simulate the appearance of the unsafe place with a stored token. This means that there is an unsafe state in the workflow of the proposed protocol. The unsafe state will be discussed in the next chapter.

3.3 Verification phase

The purpose of this phase is to verify the protocol using the tools provided in CPN Tools. The outcome of this phase consists of the simulation results for each scenario and generated state space reports for each simulation. These results will be described later in the chapter 4.

3.4 Implementation phase

After protocol verification using formal method tool in the previous phase, the prototype software has been implemented in order to validate the correctness of the protocol's workflow. The implemented software demonstrates the 2-step authentication process to grant an access to the data storage device.

3.4.1 Overview

The implemented software contains two steps of authentication process. At the first step, the user has to input the registered username and password. Then the system will show additional user interface window for the user to input the set of 6-digits number as the second step of authentication. The set of number for input in this step will be generated from the system and display in another user interface window. The user has to complete both steps, in order to gain access to the system.

3.4.2 Declarations

The variables that will be used in the implemented software have to be unique, because the purpose of the software is to prevent the usage and control the access of USB data storage device from improper uses.

```
public class MainLogin extends javax.swing.JFrame {
String userdefined = "admin";
String passdefined = "1234";

    /**
     * Creates new form MainLogin
     */
    public MainLogin() {
        initComponents();
    }
```

Figure 3.32 Username and Password declaration

```
public static String LoginUsername;
public static String LoginPassword;
public static String usbserial = "fa37d330a8c25480892c1e6bfaae1b9b";
public static String imeiserial = "5787011053026016";
public static Timer timer;
```

Figure 3.33 USB serial and IMEI serial declaration

The Figure 3.32 and 3.33 show list of the variable which is used in the software. Username and Password are the parameters belonged to individual users, while IMEI number and USB serial number are variable number related to each device.

By using the USB unique number (*usbserial*) and Smartphone IMEI (*imeiserial*) as the factor in OTP generation, the generated OTP can be used with specified USB device and Smartphone which have the same value in the source code.

3.4.3 Algorithms

Most of the algorithms in the implemented software are the conditions such as if-else loops and switch, which control the access of the process that initiated by the user. The hash function is also applied for digest the string, and the timer function is used to control the access and limit the time usage.

Login Process When compiles the source code, this is the first step in authentication process. The client has to input his username and password to proceeds to the next step. Incorrect username or password or both of them will not be granted an access.

```
//check if the input is valid
        if (userlogin.compareTo(userdefined)!=0 || (passlogin.compareTo(passdefined)!=0))
            {
            JOptionPane.showMessageDialog(null, "Incorrect Username or Password", "Warning",JOptionPane.WARNING_MESSAGE);
            jTextField1.setText("");
            jPasswordField1.setText("");
            }
        else if (userlogin.compareTo(userdefined)!=1 && (passlogin.compareTo(passdefined)!=1))
        {
//Username & Password is correct, Send values to create OTP
            try {
                AuthenProtocol.LoginUsername = jTextField1.getText();
                AuthenProtocol.LoginPassword = jPasswordField1.getText();
                AuthenProtocol.main();
                } catch (Exception ex) {
                Logger.getLogger(MainLogin.class.getName()).log(Level.SEVERE, null, ex);
                }
```

Figure 3.34 Username and password verification

OTP Generation The factors which are used in the OTP generation process consist of; Username (*LoginUsername*), Password (*LoginPassword*), USB unique serial number (*usbserial*), IMEI serial number (*imeiserial*), Timestamp (*timestamp*) and Random Nonce (*random_nonce*). These factors are concatenate together and sent to SHA-1 hash function. The result is in the form of 6-digits number.

```
//Concatenate String : Username + Password + USB_SERIAL + IMEI_SERIAL + TIMESTAMP + Random Nonce
String session = LoginUsername;
session = session.concat(LoginPassword);
session = session.concat(usbserial);
session = session.concat(imeiserial);
session = session.concat(timestamp);
session = session.concat(random_nonce);

//Generate otp
shalotp  = shaihash(session);
```

Figure 3.35 Factors concatenation

OTP Verification Verifying the input OTP from the client is simple, if the input OTP is equals to another one in the system then the client is granted. If they are not equals, then the client needs to re-input the OTP again. The problem is how to know that the OTP is already used before. To solve this problem, we declare the integer variable *result_check* in order to verify the OTP that is already used or not (Figure 3.6). After verifying the use of OTP finished, the result will be displayed in the pop-up message dialog box as shown in Figure 3.37.

```
if ((GetServerOtp.compareTo(GetUserOtp)==0) && (result_check == 0))
{
    OTP.ReturnResult(1);
    result_check = 0;
}
else if ((GetServerOtp.compareTo(GetUserOtp)!=0) && (result_check != 0))
{
    OTP.ReturnResult(2);
    GetUserOtp = "";
}
else
{
    OTP.ReturnResult(2);
    GetUserOtp = "";
}
```

Figure 3.36 Conditions to verify if OTP corrected and was already used or not

```
public static void ReturnResult(int n)throws Exception{

    int result = n;

    if (result == 1){
        //If the OTP is matched
        JOptionPane.showMessageDialog(null, "Access Complete", "Access Granted",JOptionPane.INFORMATION_MESSAGE);
        System.exit(0);

    }else if(result == 2){
        //If the OTP is NOT matched
        JOptionPane.showMessageDialog(null, "Abort : The OTP is incorrect", "Access Denied",JOptionPane.ERROR_MESSAGE);

        //Check number of attempts. If more than Designated times, then exit.
        Attempt();
    }
}
```

Figure 3.37 Result of verification shows in message dialogbox

Countdown Timer Each generated OTP has life time limitation. The OTP is valid only 30 seconds after being generated. Figure 3.35 shows the algorithm of countdown timer for each OTP. The timer starts when the user requests the OTP and counts from 30 seconds to 0. If the timer reaches the 0, then the current OTP is invalid and the user has to request the new OTP.

```
timer.restart();
 <snip>

if(req<req_limit)
{
    <snip>

        elapsedSeconds =  AccessControl.SEC;
        timer.start();
    <snip>
}
else
{
    JOptionPane.showMessageDialog(null, "The request is reached the limitation", "Limit Exceeded",JOptionPane.ERROR_MESSAGE);
    System.exit(0);
}
```

Figure 3.38 Countdown timer algorithm

Rate-limit The rate-limit is the limitation which allows the user inputs the OTP in a period of time. In this software, the user can attempt the input three times per one OTP. If the limitation is reached, then the system will shutdown itself.

```
public static Integer atmp = 0;
public static Integer limit_attempt = 3;  // number of time user can attempt.
```

Figure 3.39 Rate limitation declarations

```
public static int Attempt(){
    if (atmp <(limit_attempt-1)){
        atmp++;
    }else{
        System.exit(0);
    }
    return 0;
}
```

Figure 3.40 Rate limitation conditions

Request-limit The request-limit allows the user to request the new OTP only 3 times. If the user request the new OTP more than three times, the system will shutdown itself.

```
Integer req = 0, req_limit = 3;
```

Figure 3.41 Request limitation declarations

```
if(req<req_limit)
{
  ...
}
else
{
    JOptionPane.showMessageDialog(null, "The request is reached the limitation", "Limit Exceeded",JOptionPane.ERROR_MESSAGE);
    System.exit(0);
}
```

Figure 3.42 Request limitation conditions

3.5 Validation phase

The purpose of this phase is to compile investigate the results of the implemented software from many action which may be occurs. Then the implemented software will be discussed about the security properties which were provided. This will be described in the chapter 4.

3.6 Analysis phase

The purpose of this phase is to compile analyze the result of the proposed protocol model from the simulations and the implemented software in order to find the result from both model and software are achieved the goal. The security properties of the proposed model and implemented software will also be analyzed and studied for the improvement. This phase will be described in the chapter 5.

CHAPTER IV
EXPERIMENT RESULT

4.1 The CPN model result

From the previous chapter, we implemented model and created scenarios of CPNs model for the proposed protocol, in order to ensure that the proposed protocol is able to handle specified attacks from the intruder. The first scenario is normal workflow of the protocol. In second scenario, the intruder attempts to change Diffie-Hellman key exchange parameter in order to intercept the protocol. And the last scenario, the intruder tries harder to intercept the protocol by changing Diffie-Hellman key exchange parameter and USB unique serial number.

4.1.1 Scenario 1

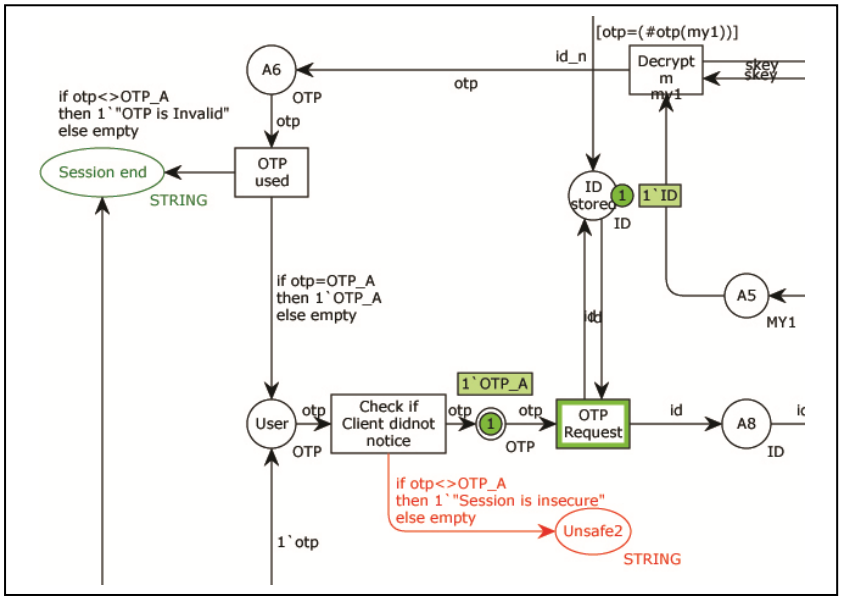

Figure 4.1 A state from scenario 1

CPN Tools state space report for:
/cygdrive/C/Users/Siripoom/Downloads/Coloured Petri Nets/Prototype_001_Man in the middle_007.cpn
Report generated: Tue Apr 7 22:56:39 2015

Statistics

State Space
Nodes: 50
Arcs: 49
Secs: 0
Status: Full

Scc Graph
Nodes: 50
Arcs: 49
Secs: 0

Boundedness Properties

Best Integer Bounds

	Upper	Lower
Client'Unsafe2 1	0	0
Intruder'Capa 1	1	1
Intruder'Capa1 1	1	1
Intruder'Intruder_get_K_AB 1	0	0
Intruder'K_AZ 1	0	0
Intruder'K_BZ 1	0	0
Intruder'USBx 1	0	0
Intruder'USBz 1	1	1
Intruder'Unsafe1 1	0	0
Intruder'gx 1	0	0
Intruder'gy 1	0	0
Intruder'gz 1	1	1
Server'New_OTP 1	1	0
Server'Reject 1	0	0
Server'Seed 1	1	0
Server'Store_ID 1	1	0
Server'Store_OTP 1	1	0
Server'Store_Seed 1	1	0
Server'Unsafe3 1	0	0
Server'Y 1	1	1

Figure 4.2 State Space Report from scenario 1

```
Best Upper Multi-set Bounds
  Client'Unsafe2 1    empty
  Intruder'Capa 1     1`0
  Intruder'Capa1 1    1`0
  Intruder'Intruder_get_K_AB 1empty
  Intruder'K_AZ 1     empty
  Intruder'K_BZ 1     empty
  Intruder'USBx 1     empty
  Intruder'USBz 1     1`USB_Z
  Intruder'Unsafe1 1  empty
  Intruder'gx 1       empty
  Intruder'gy 1       empty
  Intruder'gz 1       1`gz
  Server'New_OTP 1    1`OTP_A
  Server'Reject 1     empty
  Server'Seed 1       1`SEED_SV
  Server'Store_ID 1   1`ID
  Server'Store_OTP 1  1`OTP_A
  Server'Store_Seed 1 1`SEED_SV
  Server'Unsafe3 1    empty
  Server'Y 1          1`gy

Best Lower Multi-set Bounds
  Client'Unsafe2 1    empty
  Intruder'Capa 1     1`0
  Intruder'Capa1 1    1`0
  Intruder'Intruder_get_K_AB 1 empty
  Intruder'K_AZ 1     empty
  Intruder'K_BZ 1     empty
  Intruder'USBx 1     empty
  Intruder'USBz 1     1`USB_Z
  Intruder'Unsafe1 1  empty
  Intruder'gx 1       empty
  Intruder'gy 1       empty
  Intruder'gz 1       1`gz
  Server'New_OTP 1    empty
  Server'Reject 1     empty
  Server'Seed 1       empty
  Server'Store_ID 1   empty
  Server'Store_OTP 1  empty
  Server'Store_Seed 1 empty
  Server'Unsafe3 1    empty
  Server'Y 1          1`gy
```

Figure 4.2 State Space Report from scenario 1 (cont.)

When there is no attack from intruder in this scenario, the result of protocol in this scenario is similar to normal workflow. Since the intruder allows the message pass without modification, he cannot decrypt the encrypted message which contained OTP. This is because the properties of Diffie-Hellman key exchange method that the intruder has to modify the message from both sides to get Diffie-Hellman key

exchange parameters and generate the session key. In figure 4.1 shows a state of the protocol workflow during simulating the scenario 1 that the OTP token is *OTP_A* which is for the client *A* to use. The State Space Report in Figure 4.2 shows that there is no token occurred in *Unsafe1*, *Unsafe2*, and *Unsafe3* places which mean that the protocol is valid in the normal situation. The upper and lower integer bounds of *USBx*, *USBz*, *gx*, *gy*, and *gz* place in the intruder side are not change. We can confirm there is no interception from the Intruder's places which are showed "*empty*" in *Best Upper Bounds* and *Best Lower Bounds* topics.

4.1.2 Scenario 2

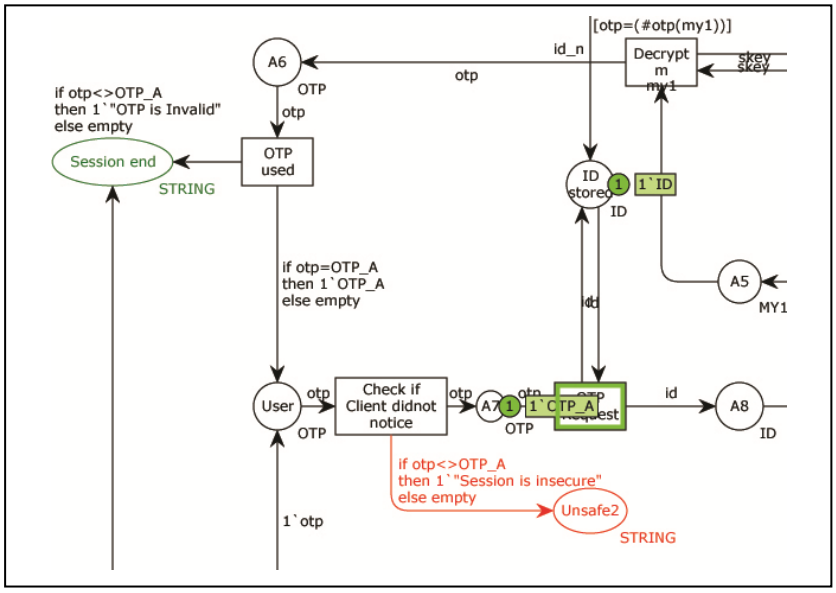

Figure 4.3 Client side in scenario 2

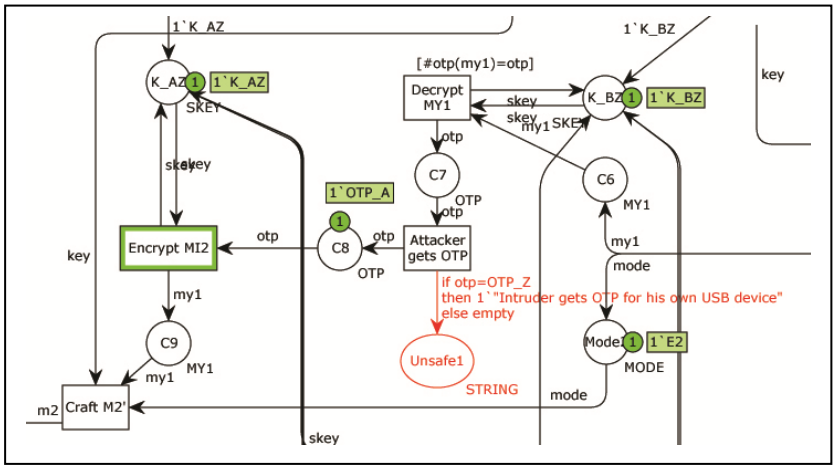

Figure 4.4 Intruder side in scenario 2

In this scenario, both server and client do not notice man-in-the-middle attack from the intruder. The client can still use the retrieved *OTP_A* token, and the server is able to generate the new OTP for each request. However, the intruder cannot use an *OTP_A*, as shown in the Figure 4.4 because the OTP generation algorithm requires the unique serial number of USB data storage device as one of the factor which makes *OTP_A* useless to the intruder. Figure 4.4 demonstrates that the intruder gets the Diffie-Hellman key exchange parameters *gx* and *gy* by tapping and rerouting the messages but he cannot use the OTP which is retrieved from the decrypted message.

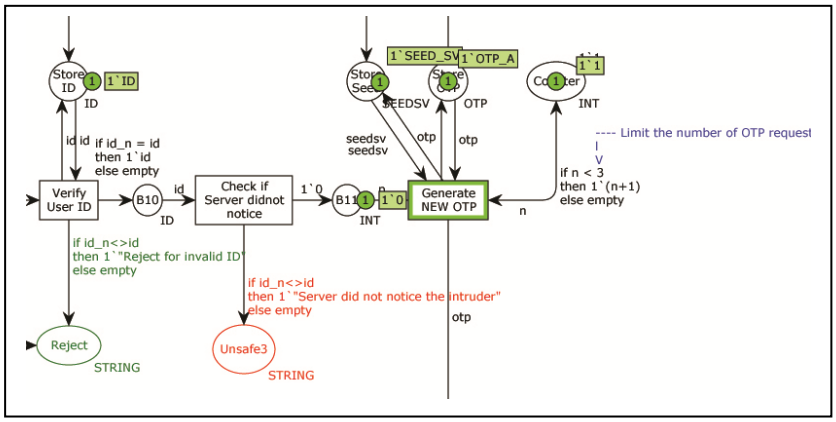

Figure 4.5 Server side in scenario 2

CPN Tools state space report for:
/cygdrive/C/Users/Siripoom/Downloads/Coloured Petri Nets/Prototype_001_Man in the middle_007.cpn
Report generated: Wed Apr 8 00:07:40 2015

Statistics

State Space
Nodes: 538
Arcs: 591
Secs: 0
Status: Full

Scc Graph
Nodes: 538
Arcs: 591
Secs: 0

Boundedness Properties

Best Integer Bounds

	Upper	Lower
Client'Unsafe2 1	0	0
Intruder'Capa 1	1	0
Intruder'Capa1 1	1	0
Intruder'Intruder_get_K_AB 1	1	0
Intruder'K_AZ 1	1	0
Intruder'K_BZ 1	1	0
Intruder'USBx 1	0	0
Intruder'USBz 1	1	1
Intruder'Unsafe1 1	0	0
Intruder'gx 1	1	0
Intruder'gy 1	1	0
Intruder'gz 1	1	0
Server'New_OTP 1	1	0
Server'Reject 1	0	0
Server'Seed 1	1	0
Server'Store_ID 1	1	0
Server'Store_OTP 1	1	0
Server'Store_Seed 1	1	0
Server'Unsafe3 1	0	0
Server'Y 1	1	1

Best Upper Multi-set Bounds
Client'Unsafe2 1 empty
Intruder'Capa 1 1`0
Intruder'Capa1 1 1`0
Intruder'Intruder_get_K_AB 1 1`K_AB
Intruder'K_AZ 1 1`K_AZ
Intruder'K_BZ 1 1`K_BZ

Figure 4.6 State Space Report from scenario 2

```
   Intruder'USBx 1     empty
   Intruder'USBz 1     1`USB_Z
   Intruder'Unsafe1 1  empty
   Intruder'gx 1       1`gx
   Intruder'gy 1       1`gy
   Intruder'gz 1       1`gz
   Server'New_OTP 1    1`OTP_A
   Server'Reject 1     empty
   Server'Seed 1       1`SEED_SV
   Server'Store_ID 1   1`ID
   Server'Store_OTP 1  1`OTP_A
   Server'Store_Seed 1 1`SEED_SV
   Server'Unsafe3 1    empty
   Server'Y 1          1`gy

Best Lower Multi-set Bounds
   Client'Unsafe2 1    empty
   Intruder'Capa 1     empty
   Intruder'Capa1 1    empty
   Intruder'Intruder_get_K_AB 1empty
   Intruder'K_AZ 1     empty
   Intruder'K_BZ 1     empty
   Intruder'USBx 1     empty
   Intruder'USBz 1     1`USB_Z
   Intruder'Unsafe1 1  empty
   Intruder'gx 1       empty
   Intruder'gy 1       empty
   Intruder'gz 1       empty
   Server'New_OTP 1    empty
   Server'Reject 1     empty
   Server'Seed 1       empty
   Server'Store_ID 1   empty
   Server'Store_OTP 1  empty
   Server'Store_Seed 1 empty
   Server'Unsafe3 1    empty
   Server'Y 1          1`gy
```

Figure 4.6 State Space Report from scenario 2 (cont.)

Since the intruder changes Diffie-Hellman key exchange parameters in this scenario, he can generate the encryption key K_{AZ} for communicating with the client and K_{BZ} for communicating with the server. For this reason, the values of integer bounds of place *K_AZ* and *K_BZ*, which are the session key for communication between intruder-client and intruder-server, had been changed as shown in Figure 4.6. The place *gx*, *gy*, and *gz* in the State Space Report which are Diffie-Hellman key agreement parameters also had been changed due to the interception and alteration from the intruder. The upper and lower integer bounds of *Intruder_get_K_AB* place are changed as well since the intruder is able to create the *K_AB* from the *gx* and *gy*

token, but it is useless since the intruder has to use *K_AZ* and *K_BZ* instead. The integer bounds of *USBx* and *USBz* place are unchanged, since the intruder left the USB serial number unchanged. Finally, all of the *Unsafe* places (*Unsafe1*, *Unsafe2*, and *Unsafe3*) still have no token occurs. As a result, even the intruder had intercepted the communication; the protocol is still achieving the goal of security on the OTP-level.

4.1.3 Scenario 3

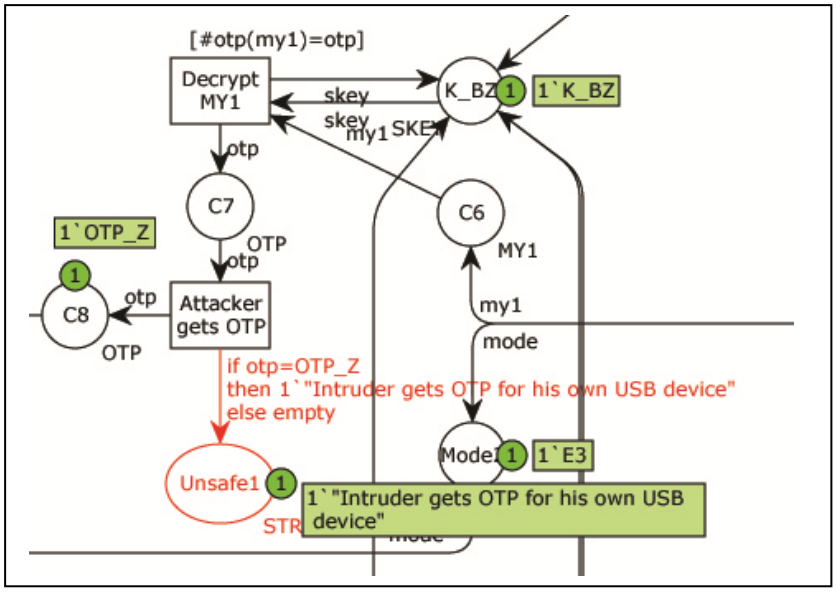

Figure 4.7 The unsafe state in the scenario 3

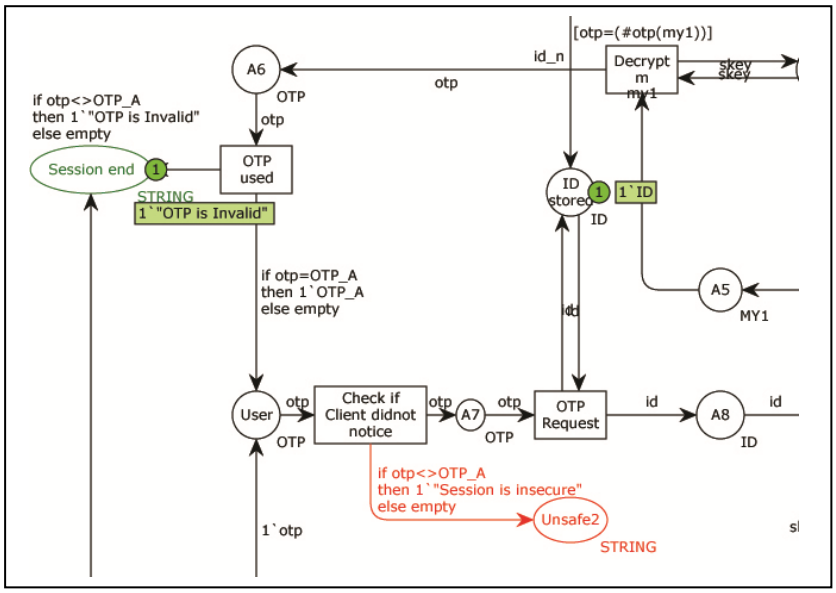

Figure 4.8 Session end after client notices the intruder

In this scenario, the intruder succeeds to obtain *OTP_Z* token (shown in Figure 4.7) for his own USB data storage device. Also, the '*Unsafe1*' place, which is used to confirm that the unsafe state can occurred during workflow, shows that there is a token stored in this place after the simulation finished. Although, the client is able to notice the attack and suspend the session (Figure 4.8), the *OTP_Z* which is generated by the server does not belong to a client's USB data storage device. The protocol is still vulnerable to this kind of attack since an intruder is able to obtain OTP and gains access to services. Anyways, the intruder obtains the *OTP_Z* token, not *OTP_A*, which indicates that the intruder is still unable to gain an access to the client's USB device (USB_A).

From the State Space report in Figure 4.9, the values of integer bounds of place *K_AZ*, *K_BZ*, *USBx*, *USBz*, *gx*, *gy*, and *gz* had been changed because the intruder intercepted and replaced both Diffie-Hellman key agreement parameters and USB unique serial number in this scenario. The integer bound values of *Unsafe1* place in the intruder model have changed because there is the token occurs, which means this simulation environment of the protocol model had an insecure state. However, the other *Unsafe* places (*Unsafe2* and *Unsafe3*) have no changes in the integer bounds because the communication had been terminated already. The *Session End* place at the client side also has "*OTP is invalid*" token represents that the user was able to retrieve the OTP, but the client cannot use this OTP on his USB device. Because the OTP was generated from altered USB serial number which was replaced by the intruder.

CPN Tools state space report for:
/cygdrive/C/Users/Siripoom/Downloads/Coloured Petri Nets/Prototype_001_Man in the middle_007.cpn
Report generated: Wed Apr 8 00:10:58 2015

Statistics

State Space
Nodes: 154
Arcs: 207
Secs: 0
Status: Full

Scc Graph
Nodes: 154
Arcs: 207
Secs: 0

Boundedness Properties

Best Integer Bounds

	Upper	Lower
Client'Unsafe2 1	0	0
Intruder'Capa 1	1	0
Intruder'Capa1 1	1	0
Intruder'Intruder_get_K_AB 1	1	0
Intruder'K_AZ 1	1	0
Intruder'K_BZ 1	1	0
Intruder'USBx 1	1	0
Intruder'USBz 1	1	0
Intruder'Unsafe1 1	1	0
Intruder'gx 1	1	0
Intruder'gy 1	1	0
Intruder'gz 1	1	0
Server'New_OTP 1	0	0
Server'Reject 1	0	0
Server'Seed 1	1	0
Server'Store_ID 1	1	0
Server'Store_OTP 1	1	0
Server'Store_Seed 1	1	0
Server'Unsafe3 1	0	0
Server'Y 1	1	1

Best Upper Multi-set Bounds
Client'Unsafe2 1 empty
Intruder'Capa 1 1`0
Intruder'Capa1 1 1`0
Intruder'Intruder_get_K_AB 1 1`K_AB
Intruder'K_AZ 1 1`K_AZ
Intruder'K_BZ 1 1`K_BZ
Intruder'USBx 1 1`USB_A
Intruder'USBz 1 1`USB_Z
Intruder'Unsafe1 1 1`"Intruder gets OTP for his own USB device"
Intruder'gx 1 1`gx

Figure 4.9 State Space Report from scenario 3

```
   Intruder'gy 1       1`gy
   Intruder'gz 1       1`gz
   Server'New_OTP 1    empty
   Server'Reject 1     empty
   Server'Seed 1       1`SEED_SV
   Server'Store_ID 1   1`ID
   Server'Store_OTP 1  1`OTP_Z
   Server'Store_Seed 1 1`SEED_SV
   Server'Unsafe3 1    empty
   Server'Y 1          1`gy

Best Lower Multi-set Bounds
   Client'Unsafe2 1    empty
   Intruder'Capa 1     empty
   Intruder'Capa1 1    empty
   Intruder'Intruder_get_K_AB 1  empty
   Intruder'K_AZ 1     empty
   Intruder'K_BZ 1     empty
   Intruder'USBx 1     empty
   Intruder'USBz 1     empty
   Intruder'Unsafe1 1  empty
   Intruder'gx 1       empty
   Intruder'gy 1       empty
   Intruder'gz 1       empty
   Server'New_OTP 1    empty
   Server'Reject 1     empty
   Server'Seed 1       empty
   Server'Store_ID 1   empty
   Server'Store_OTP 1  empty
   Server'Store_Seed 1 empty
   Server'Unsafe3 1    empty
   Server'Y 1          1`gy
```

Figure 4.9 State Space Report from scenario 3 (cont.)

In conclusion, we can ensure that the proposed protocol is able to tolerance the Man-in-the-Middle attack from the intruder. However, there is some vulnerable in the model that let the intruder be able to obtain the *OTP_Z* token, that can be used with the USB_Z device which is possessed by the intruder. Besides, the *OTP_Z* token is can be used with the USB_Z device only, the USB_A device is still safe from the attack and the intruder is unable to access the target USB device.

4.2 The implemented Software result

After protocol verification using formal method tool, the source code has been implemented in order to validate the correctness of the protocol's workflow. The

implemented software demonstrates the 2-step authentication process to grant an access to the data storage device. In this thesis, we implement the prototype software using JAVA programming language and the *NetBeans IDE 8.0.2* as a compiler. The reason of implementing the software using JAVA is that we can design the user interfacing with graphical wizard tool.

Figure 4.10 MainLogin User interface

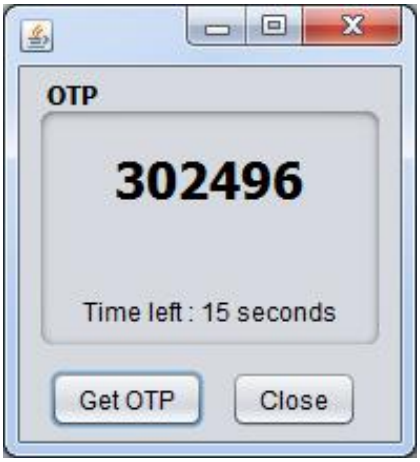

Figure 4.11 OTP-Display User interface

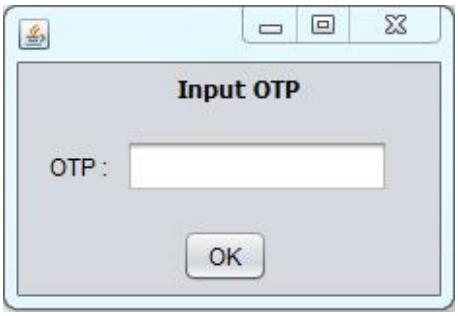

Figure 4.12 OTP-Input User interface

The software has three GUIs (graphic user interface) which are MainLogin, OTP-Display, and OTP-Input (Figure 4.10 to 4.12). When the program starts, the MainLogin interface appears and client has to input username and password as a first step of authentication. If the client inputs wrong username or password or both, the system dialog box will appear and notice the client that the inputs are incorrect as shown in Figure 4.13.

Figure 4.13 Warning message dialog box

After client inputs the valid username and password, two additional interfaces will appear. The OTP-Display user interface appears with 6-digits number and 30-seconds countdown timer. Another interface is OTP user interface which displays a textbox window for entering the set of correct number. In this step, the client has to put the 6-digit number, which is shown in OTP-Display user interface, into the empty textbox on the OTP-Input user interface to complete the second step of authentication. If the client inputs the OTP from the OTP-Display user interface into the OTP-Input user interface correctly, the client will gain an access to the system legally (Figure 4.14).

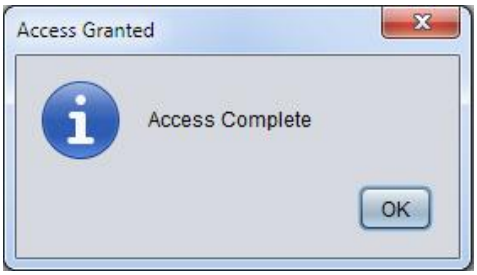

Figure 4.14 Access granted dialog box

However, the client has only 30 seconds to enter the correct OTP before the current OTP will be expired (Figure 4.15), this condition will prevent the system from offline-password guessing attack. After the current OTP is invalided, the client needs to get the new set of number by click "Get OTP" button in the OTP-Display user interface. Moreover, client can attempt the input only three times per one OTP. If the client cannot input the OTP correctly in three times, the system will shutdown itself automatically. This is called "*Rate-limiting*" which is used to prevent the system from the brute-force attack and offline-guessing attack.

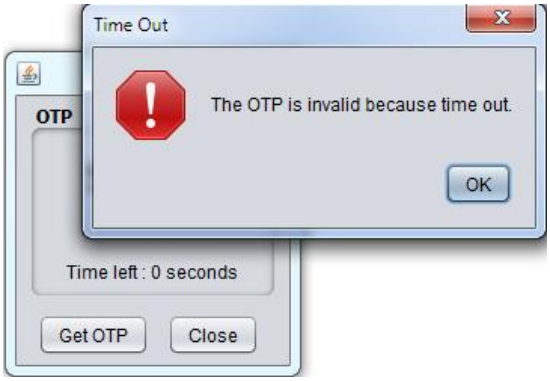

Figure 4.15 Time-out dialog box

Another prevention feature is called "*Request-limiting*", which allows the user to request the new set of OTP for three times only. If the user requests the forth OTP, the warning dialog box will appear as shown in Figure 4.16. This feature is used to prevent the system from the user flooding the request message to the server which may lead to DOS attack.

Figure 4.16 Request-limit dialog box

Username and Password are the parameters belonged to individual users, while IMEI number and USB serial are variable number related to each device. By using the USB unique number (*Usb serial*) and Smartphone IMEI as the factor of OTP generation, the generated OTP can be used with specified USB device and Smartphone which have the same value in the source code.

From the reasons above, only a legitimated client, who possesses the specified Smartphone and USB data storage device and passes the registration procedure appropriately, is able to access the data stored in the USB storage device. Anyone who did not registered and/or possessed the specified Smartphone and USB device, is unable to gain the permission to access the device.

Finally, the Figure 4.17 shows the flowchart process of the implemented software. The implemented software workflow starts with the appearance of MainLogin window. After user inputs the username and password correctly, OTP-Display and OTP-Input window will appear. The user has to input the OTP, which is six-digit number shows in OTP-Display window, in order to gain an access. If the user inputs incorrect OTP more than three times, the software will shutdown itself. If the OTP is used already or OTP is timed out, the user has to request the new OTP or the software will shutdown itself too. Finally, if the user had request the new OTP three times already, the software will also shutdown itself.

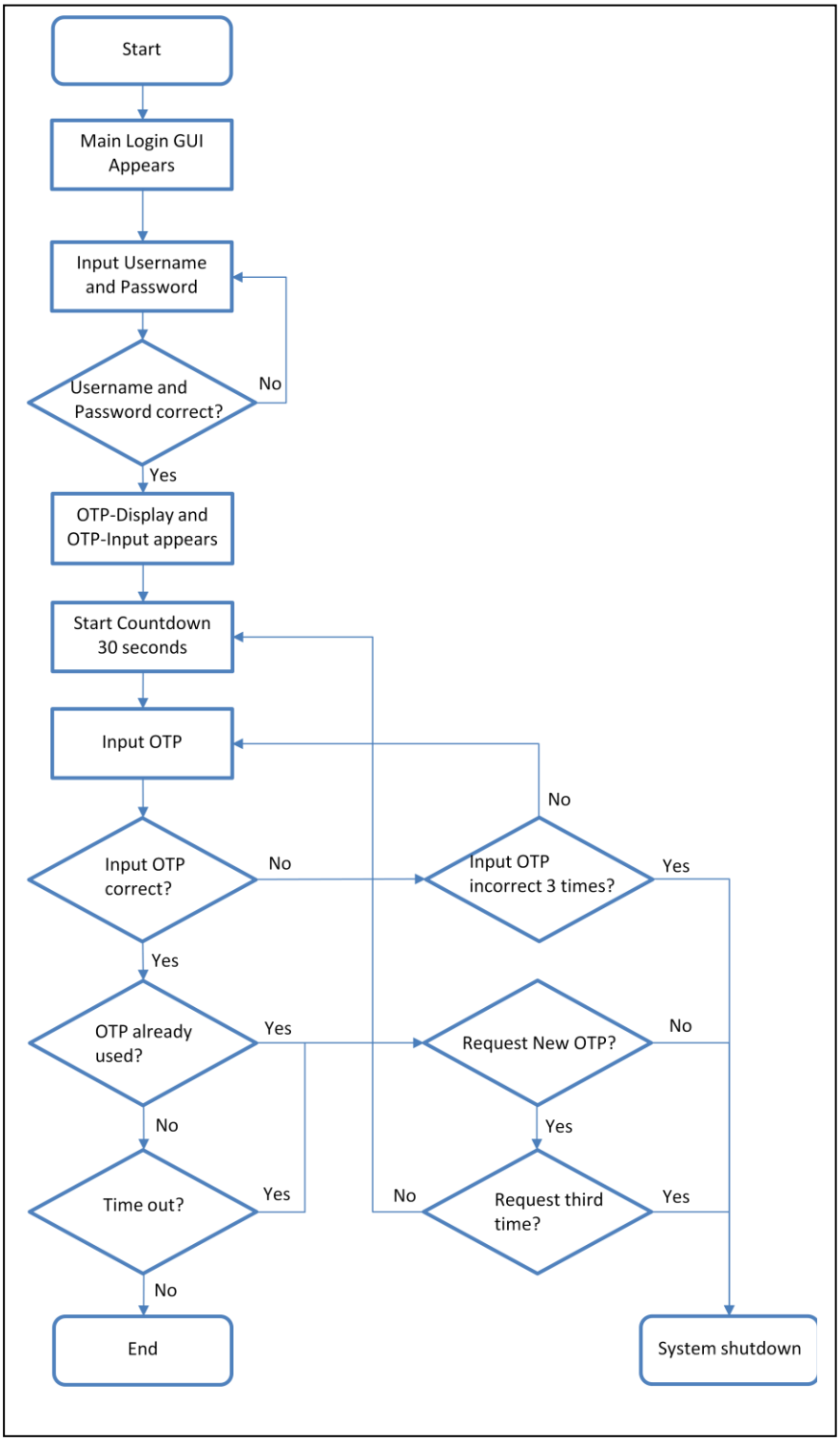

Figure 4.17 Flowchart of implemented software

CHAPTER V
ANALYSIS

In this chapter, we present a protocol analysis to ensure that our proposed protocol model and implemented software, both of them provide basic security properties much enough to achieve our defined goals.

5.1 Protocol model analysis

This subsection analyses the security properties of the proposed protocol by running the simulation model which were built in CPNs tool.

5.1.1 Integrity

Having an integrity property will ensure the original of the exchanged message between two parties. If the message had been changed, the protocol participants should be able to detect the changes and reject the message. Since our protocol applies only symmetric encryption technique with hash function on some parameters; for example, the protocol still has enough integrity in the OTP-level as in the scenario 2. Although the intruder changed both Diffie-Hellman key exchange parameter and symmetric session key, the client still can decrypts the modified message and use the received OTP without any problem because the OTP generation process employs unique number of USB data storage device as one of the factor. However, the intruder cannot modify this parameter. If he does, the result will be another OTP for the USB device which has the modified unique number parameter.

5.1.2 Confidentiality

Confidentiality prevents unauthorized persons from accessing and reading the messages between intended participants. However, our protocol aims to have a light-weighted computation. With confidentiality requirement, the message encryption

will apply only on the important information parameters. Although the intruder can decrypt the encrypted message (as in Scenario 2), but he is still unable to use the retrieved OTP freely. For this reason, our protocol has a property of confidentiality on an OTP-level.

5.1.3 Authentication

This is the primary concern of our proposed protocol because we attempt to develop an authentication protocol for protecting USB storage devices. To generate a unique OTP for a specific USB data storage device from many identical factors, the protocol is able to provide an authentication service to both client and server using this method.

5.1.4 Replay Attack

The proposed protocol is able to prevent the OTP from an intruder. By using ID and USB's serial number, which both of them are unique, as the factors to generate a unique OTP for specified user and USB devices. If the intruder had changed ID information, then the server will realize that there is another user request for OTP. Even the intruder tries to change serial number of USB device from the original to his own device, then the server will generated OTP which is for his own USB, not for an original one. Anyways, if an intruder has changed the serial number of USB, then the client is able to detect and notice about this situation since the received OTP is different from the one which is belonged to a client.

5.1.5 Prevent unauthorized users or authorized users from unauthorized use

By using unique factor such as ID, USB unique serial number, and Smartphone IMEI code, the unique OTP is created for each set of factors. An unauthorized user is unable to use an OTP because it is only for the USB device on client's hand. Even the unauthorized user changes information, the OTP output is for the modified information set. However, there is a vulnerable occurred in the scenario 3 when an intruder replaces Diffie-Hellman key exchange parameter g^x with g^z and replaces USB device unique number USB_A with USB_Z in the message *M1*. In the

message *M2*, the intruder can retrieve the OTP_Z by decrypting the OTP encryption because the OTP_Z was generated from the USB_Z owned by the intruder. This situation causes a client to notice about the existence of intruder in the protocol, so a client will terminate the session. Nonetheless, this situation may occur if the registration database system uses the "*one person-many devices*" scheme or the intruder possesses the stolen USB device and there is no update on the USB device status. If the administrator uses "*one person-one device*" database scheme or update the status of the USB device, then the situation can be prevented from occur.

5.2 Implemented Software analysis

This subsection considers the security properties of the implemented software resulting from the previous chapter.

5.2.1 Prevent unauthorized user from access

Because the OTP is generated from many unique factors which are belonged to specified user and device. *Username* and *Password* are belonged to the registered client, USB *unique serial number* is belonged to the specific USB device, and *IMEI* is belonged to specific Smartphone. The result shows that generated OTP can be used by the legitimate user who possesses the specific USB device and Smartphone. Consequently, the other users who did not registered cannot use the OTP to access the device.

5.2.2 Password guessing and Brute-Force attack

The implemented software can prevent USB device from password guessing and brute-force attack from the intruder. Because our OTP was generated randomly from many unique factors, the output OTP is impossible to calculate in the 30 seconds. Moreover, the generated OTP is in 6-digits form, and "*rate-limiting*" property which allows the user can attempt to input the password only three times per one OTP. This makes the intruder cannot attack the system using the brute-force

method to find the correct OTP. Subsequently, the implemented software is safe from such the attacks.

5.2.3 Replay attack

The implemented software code has protected the used OTP from being re-used. This property will be activated after the client inputs the correct OTP and gains the access, then the client or anyone cannot reuse the used OTP again. Similarly, the intruder cannot use the OTP to launch the replay attack on the system. Thus, the implemented software is always safe from the replay attack.

5.2.4 Prevent duplication of software from the device to others

Because the OTP is generated by using many unique factors such as *Username*, *Password*, *USB serial number*, and *Smartphone IMEI* , which made OTP is only for specific device. Even the intruder duplicates the software and data from the USB device to others, the data still cannot be accessed. This is because the software was bound to the serial number of the former USB device. Duplicating the software cannot change this specific parameter to other value.

5.3 Computational Cost Comparison

In this subsection, the proposed protocol (Fig 3.3) is compared with Yang et al. protocol [5], Lee et al. protocol [6], and He et al. [7] in term of computational cost. We divide and analyze both protocols' operation in crypto-operations term. Then, the protocols were computed using the relative computational times which were calculated as before in [24]. The notations are as follows:

T_{hash} : Computational cost of one-way hash function
T_{sym} : Computational cost of symmetric key cryptography
T_{pm} : Computational cost of elliptic curve point multiplication
T_{me} : Computational cost of modular exponentiation operation
T_{fe} : Computational cost of fuzzy extractor

To calculate precisely, it requires 0.00032s to execute the one-way hash function, 0.0056s to execute the symmetric encryption/decryption, 0.0171s to execute an elliptic curve point multiplication and 0.0192s to execute a modular exponential. The computational cost comparison shows in the Table 5.1. Since the random operation has execution time much less than hash function operation, we assume that it spends 0 second and is excluded from the calculation. From the Table 5.1, Yang et al. protocol has execution time more than other protocols in both User/Client side and Authentication Server side. The second most execution time protocol is He et al. protocol following by Lee et al. protocol. The proposed protocol has the least execution time in the comparison due to the less number of crypto-operations.

It should be noted that the T_{fe} is only available in He et al. [7] and assumed that the computational cost is the same as the elliptic curve point multiplication.

Table 5.1 Computational cost comparison table

Protocol	User/Client	Authentication Server	Total
Yang et al. [5]	$3T_{hash}+1T_{sym}+4T_{me}$ ≈ 0.08336s	$3T_{hash}+1T_{sym}+6T_{me}$ ≈ 0.12176s	$6T_{hash}+2T_{sym}+10T_{me}$ ≈ 0.20512s
He et al. [24]	$5T_{hash} + 1T_{sym} + 2T_{pm}$ $+ T_{fe} \approx 0.0585$s	$4T_{hash} + 1T_{sym} + 2T_{pm}$ ≈ 0.04108s	$9T_{hash} + 2T_{sym} + 4T_{pm}$ $+ T_{fe} \approx 0.09958$s
Lee et al. [23]	$5T_{hash} + 1T_{sym} + 2T_{pm}$ ≈ 0.0414s	$4T_{hash} + 1T_{sym} + 2T_{pm}$ ≈ 0.04108	$9T_{hash}+2T_{sym} +4T_{pm}$ ≈ 0.08248s
Proposed Protocol	$1T_{hash} + 1T_{sym} + 2T_{me}$ ≈ 0.04432s	$2T_{hash} + 1T_{sym} + 2T_{me}$ ≈ 0.04464s	$3T_{hash} + 2T_{sym} + 4T_{me}$ ≈ 0.08896s

The comparison from the table shows that the proposed protocol has less execution time than Yang et al. [5] and He et al.[7] but slightly more than Lee et al.[6] protocol. However, both Lee et al.[6] and He et al. [7] protocol are inappropriate for using in the digital evidence acquisition process. This is because the AS needs to verify inputs from the user via the PC which means many operations will be computed in the PC and altered the data in RAM.

All data and running process which were stored in RAM are very important to Digital evidence acquisition process. These data should not be altered or keep the changes as little as possible during the process due to the trust of evidence acquisition. Our proposed protocol aims to minimize the computation process in the target PC, so we use the Smartphone as an auxiliary device to communication with the AS. This makes our proposed protocol more appropriate for digital evidence acquisition process than other.

5.4 Comparison of the proposed technique with Google Authenticator

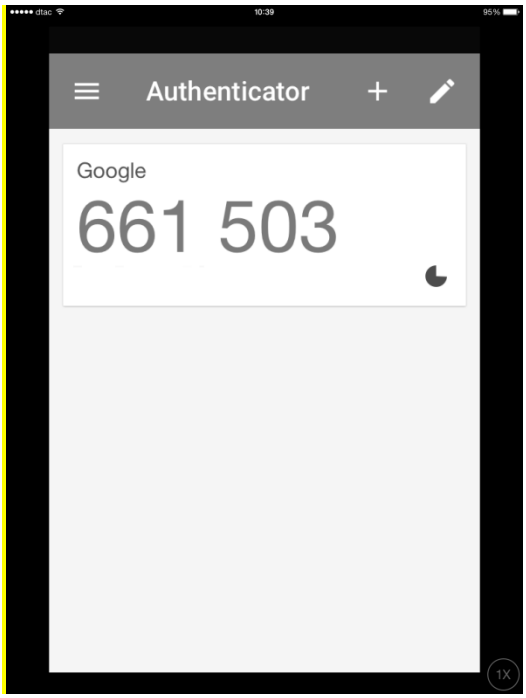

Figure 5.1 Google authenticator application on iPad (iOS)

"*The Google authenticator is the application implements TOTP (Time-based One-time Password) security tokens from RFC 6238 in mobile apps made by Google*" (Wikipedia.org)

The google authenticator provides 6-digit one-time password for user to log into the Google services. The application can also generates codes for third-party application too. Normally, users have to install the application on their Smartphone.

When log into services or sites that need two-factor authentication, such as Google mail service, the user has to input the user account and password to the site and the application provide the 6-digit OTP. When the user inputs the correct number, the site will authenticate the user and give the permission to access.

If any intruder wants to access the service, he has to know the shared secret of access the device which is installed the application physically. Another way to attack is man-in-the-middle attack: if the user's computer has been infected, then the attack will be able to obtain and alter the username, password and OTP to gain the permission to access. Google authenticator is available for many operating system platforms: iOS, Andriod, and Blackberry. Also many third party implementations are available too.

The Google authenticator also has the countdown timer to limit the lifetime of each generated OTP. The small circle on the right will disappear slowly which equal to 30 seconds approximately. When the small circle disappears, the OTP set will change into new one.

Since the application does not have any button, so the user has to wait until the timer reset in order to obtain the new OTP. It implies that the application does not need the "*request-limiting*" feature to control the request message at the server. The reason is that the server possibly does not want to handle the request message from many clients, which might be thousand or million request messages, in the same time.

The source code of the application in the original version does not have the "*rate-limiting*" feature. The clients can attempt the wrong OTP as many times as they want before timing out. This vulnerable may lead the intruder who be able to launch the brute-force attack the 6-digit number of OTP. If the attack processes on supercomputer, then the OTP can be exposed. However, the application becomes an open source software, which means the user can edit the source code of the application to whatever they want. In the case, the user can add the "*rate-limiting*" feature by adding the code to limit the number of OTP attempts before the OTP status is invalid. In addition, most of the user leaves the source code of the application as the original from the app store.

CHAPTER VI
CONCLUSION AND FUTURE WORKS

The main objective of this research is to design the authentication protocol in order to add the security layer on digital evidence acquisition devices. We proposed the light-weighted 2-step authentication protocol which uses only Diffie-Hellman key exchange method to create a symmetric session key and hash function. We ensured a correctness and security of the proposed protocol by using the formal method tool: the CPNs Tools to create the model of the protocol for verification. We also implemented the prototype software of the protocol for validation the process of 2-step authentication. To prove security of the protocol, we created the intruder model that is able to modify Diffie-Hellman key exchange parameter and USB device unique ID number. We constructed the intruder model and scenarios in the CPNs Tools, according to the assumptions which are given to the intruder. Then we simulated the protocol for every scenario to check the unsafe states that might be undiscovered by manual checking. The following is a summary of our work related to these proposed objectives.

1) To research and compare among several two-factor authentication protocols.

We studied the two-factor authentication protocols and Formal method tool in Chapter 2. Then we compared the advantages and disadvantages among those protocols to create the Two-factor based authentication protocol for USB digital evidence acquisition devices.

2) To develop an appropriated authentication protocol for USB type data storage.

In Chapter 3 we conducted and developed the authentication protocol which will apply on the digital evidence acquisition devices, especially. The proposed protocol model and implemented prototype software are proven to be secured for specific applications usages in Chapter 4.

3) To propose the authentication protocol that can prevent access from unauthorized users.

4) To propose the authentication protocol that can prevent the device from abused in an unassigned task and limit the access time.

5) To propose the authentication protocol that can prevent duplication of software from the device to others.

As shown in Chapter 4, our proposed authentication protocol is proven to be able to prevent the devices from any unauthorized users, prevent the device from abused in an unassigned task and limit the access time, and prevent duplication of software from the device to others. The proposed authentication protocol can prevent illegal access. Because, we used many unique factors such as *username and password*, *USB serial number*, and *Smartphone IMEI* to generates the OTP. The output OTP is only for the person who knows the correct username and password, possessed the specified data acquisition device and Smartphone. Although the intruder

6) To verify the security of purposed protocol by using formal verification tool.

The formal verification tool which was used in this research is CPN tools. The proposed authentication protocol was designed and verified using CPN tools as shown in Chapter 3 and 4, respectively. The verification result showed that the proposed protocol is proven to be able to provide basic security properties for digital evidence acquisition process.

7) To validate the functionality of the proposed protocol by implementing prototype software.

In Chapter 3 we implemented the prototype software in order to validate the functionality of the proposed authentication protocol using JAVA language. The validation result shown in Chapter 4, the implemented prototype software is proven to be able to prevent an unauthorized access as same as the result from protocol model simulation. The implemented prototype software is able to prevent not only password-guessing attack and brute-force attack, but also replay attack too.

In summary, we can ensure that our proposed protocol satisfies all basic security properties, especially in the authentication purposes. This is because the intruder cannot use the OTP_A which is only for USB_A even he can retrieves the OTP_A

by decrypting the encrypted message in the scenario 2. However, there was some vulnerable in our protocol when the scenario 3 is simulated. If the intruder replaces the unique number of USB_A with his own USB_Z in the message *M1*, then he can retrieved the OTP_Z for his own USB device by decrypting the encrypted message *M2*. The intruder now can use the retrieved OTP_Z along with his own USB for the illegal usages.

To solve this problem, we can modify the protocol by inserting a digital signature scheme. This scheme has been widely used in a research community to prevent man-in-the-middle attack effectively. By using this technique, an intruder is unable to generate signed messages using a server's private key. Both client and server have more ability to verify and authenticate each other from this technique. However, a digital signature which is an application of asymmetrical encryption algorithm introduces another trade-off to our proposed scheme. This is because a major disadvantage of asymmetrical encryption algorithm requires high computation. This weakness violates our goal to design light-weighed computation scheme that can be implemented in Smartphone or tablet, including the previous mobile phones which had low computation power. Another possible solution includes the full registration phase to register User ID, password, USB device unique number, and Smartphone IMEI to the server and stored in the database for verification.

REFERENCES

1 K. Lee, H. Yeuk, Y. Choi, S. Pho, I. You, and K. Yim, "Safe Authentication Protocol for Secure USB Memories,"Journal of Wireless Mobile Networks, Ubiquitous Computing and Dependable Applications (JoWUA) 2010; 1(1): pp. 46–55.

2 F. Y. Yang, T. D. Wu, and S. H. Chiu, "A Secure Control Protocol for USB Mass Storage Devices," IEEE Transactions on Consumer Electronics, (Volume:56 , Issue: 4), pp. 2239 – 2343, 2010

3 A. N. Magdum, and Y. M. Patil, "A Secure Data Transfer Algorithm for USB Mass Storage Devices to Protect Documents," International Journal of Emerging Engineering Research and Technology, Vol. 2, Issue 4, pp. 113-119, 2014

4 M. H. Eldefrawy, M. K. Khan, and H. Elkamchouchi, "The Use of Two Authentication Factors to Enhance the Security of Mass Storage Devices," 11th International Conference on Information Technology New Generations (ITNG), pp. 196-200, 2014

5 F. Y. Yang, T. D. Wu, and S. H. Chiu, "A secure control protocol for USB mass storage devices," IEEE Transactions on Consumer Electronics, Vol. 56, pp. 2239-2343, 2010

6 C. Lee, C. Chen, and P. Wu, "Three-factor control protocol based on elliptic curve cryptosystem for universal serial bus mass storage devices," IET Computers & Digital Techniques, vol. 7, no. 1, pp. 48-55, Jan.2013

7 D. He, N. Kumar, J.H. Lee, and R.S. Sherratt, "Enhanced three-factor security protocol for consumer USB mass storage devices, " IEEE Transactions on Consumer Electronics Vol. 60 , pp. 30-37, 2014

8 Suratose T., Napat T., and Nattawut R., "A Secure Authentication Protocol using HOTP on USB Storage Device", International Conference on Information

Science, Electronics and Electrical Engineering (ISEEE), Sapporo, Japan, April, 2014

9 W. Diffie and M. Hellman, “New directions in cryptography,” IEEE Transactions on Information Theory, Vol. 22, No. 6, pp.644-654, 1976

10 X. Jiang and J. Ling, “Simple and Effective One-Time Password Authentication Scheme,” 2nd International Symposium on Instrumentation and Measurement, Sensor Network and Automation (IMSNA), pp. 529-531, 2013

11 H. Liu and Y. Zhang, “An Improved One-time password Authentication Scheme,” 15th IEEE International Conference on Communication Technology (ICCT), pp. 1-5, 2013

12 K. Jensen, “A Brief Introduction to Coloured Petri Nets. In: E. Brinksma (ed.): Tools and Algorithms for the Construction and Analysis of Systems”, Proceeding of the TACAS'97 Workshop, Enschede, Lecture Notes in Computer Science Vol. 1217, Springer-Verlag Netherlands, pp. 203-208, 1997

13 K. Jensen, L. M. Kristensen, and L. Wells, “Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems”, International Journal on Software Tools for Technology Transfer, Springer Verlag, pp. 213-254, 2007

14 L. M. Kristensen, S. Christensen, and K. Jensen, “The Practitioner's Guide to Coloured Petri Nets”, International Journal on Software Tools for Technology Transfer, Springer Verlag, pp. 98-132, 1998

15 K. Jensen, “An Introduction to the Practical Use of Coloured Petri Nets”, W. Reisig and G. Rozenberg (eds.), Lectures on Petri Nets II: Applications, Lecture Notes in Computer Science vol. 1492, Springer-Verlag, pp. 237-292, 1998

16 Suratose T., “Protocol Engineering for Protection against Denial-of-Service Attacks”, Doctor of Philosophy Thesis, Queensland University of Technology, Australia, 2009

17 A. M. Basyouni and S. E. Tavares , “New Approach to Cryptographic Protocol Analysis using Coloured Petri Nets”, The Canadian Conf. on Electrical and Computer Engineering, pp.334-337, 1997

18 D. M. Stal, S. E. Tavares, and H. Meijer, “Backward State Analysis of Cryptographic Protocols using Coloured Petri Nets”, In Workshop on Selected Areas in Cryptography, SAC’94 Workshop Record, pp.107-118, 1994

19 K. Jensen, “Coloured Petri Nets and the Invariant-Method”, Tehoretical Computer Science, pp.317-336, 1981

20 A. Aziz and W. Diffie, “A Secure Communications Protocol to Prevent Unauthorized Access: Privacy and Authentication for Wireless Local Area Networks”, IRRR Personal Communications, pp.25-31, 1994

21 Y. Xu and X. Xie, “Modelling and Analysis of Security Protocols Using Colored Petri Nets”, Journal of computers, Vol. 6, pp.19-27, 2011

22 D. Dolev and A. Yao, “On the security of public key protocol”, 22nd Annual Symposium on Foundations of Computer Science, SFCS '81, pp.350-357, 1981

23 Gavin Lowe, “Some new attacks upon security protocols”, Proceedings of 9th IEEE Computer Security Foundations Workshop, pp.162-169, 1996

BIOGRAPHY

NAME	Mr. Siripoom Laptikultham
DATE OF BIRTH	29 January 1986
PLACE OF BIRTH	Bangkok, Thailand
INSTITUTION ATTENDED	Mahidol University, 2003-2007 Bachelor of Engineering (Computer Engineering) Mahidol University, 2011-2015 Master of Engineering (Computer Engineering)
HOME ADDRESS	99/79 Moo 4 Laddarom Elegance 5-2 Nakhon In Road Bang Khun Gong, Bang Kruai, Nonthaburi 11130 Tel. 081-2555130 E-mail : sir_lap @hotmail.com
PUBLICATION/ PRESENTATION	Siripoom Laptikultham, Modeling and Analysis of Two-factor Authentication Protocol for USB Digital Evidence Acquisition Devices, The 12th International Joint Conference on Computer Science and Software Engineering (JCSSE 2015), Faculty of Engineering and Faculty of Science at Prince of Songkla University,

HatYai Campus, Songkla, Thailand.

Siripoom Laptikultham, Two-Factor Authentication Protocol for Protecting Digital Evidence in Flash Drive, The 3rd Innovation for Crime Combating Conference and Contest 2015, May 27, 2015 , Department of Special Investigation & Mahidol University, Nakhon Pathom, Thailand.

.