บทความวิจัย

การวิเคราะห์ความมั่นคงปลอดภัยที่ได้รับการปรับปรุงให้ดีขึ้น ของโพรโตคอลสำหรับการเซ็นสัญญาที่มีความยุติธรรม ของมิคาลี่ โดยใช้คัลเลอร์เพทริเน็ต

ยงยุทธ เพิ่มพูนธนลาภ*

บทคัดย่อ

มิคาลี่ได้นำเสนอโพรโตคอลสำหรับการแลกเปลี่ยนข้อมูลอีเล็กโทรนิกส์แบบที่มีความยุติธรรม โดยใช้เทคนิคการเข้ารหัสข้อมูล ซึ่งมีชื่อว่า อีซีเอสวัน ซึ่งสามารถนำมาใช้ในการเซ็นสัญญาที่แลกเปลี่ยนกันระหว่างสองบุคคลใดๆ ต่อมาเบาว์และคณะ ค้นพบการโจมตีแบบการนำข้อความมาส่งใหม่ ในทั้งอีซีเอสวันและอีซีเอสวันที่ถูกปรับปรุงเพื่อให้มีความชัดเจนมากขึ้น หลังจากนั้นซางและคณะค้นพบการโจมตีอีกสองแบบในอีซีเอสวันที่ถูกปรับปรุงโดยเบาว์ แต่ทั้งการวิเคราะห์โดยเบาว์และซางเป็นการวิเคราะห์โดยใช้มือ ในงานที่แล้วของเราได้พัฒนาวิธีการที่เป็นแบบอัตโนมัติในการวิเคราะห์ความมั่นคงปลอดภัยในโพรโตคอลที่ใช้การเข้ารหัสข้อมูล โดยใช้คัลเลอร์เพทริเน็ต และเราได้ประยุกต์ใช้วิธีการของเราในการวิเคราะห์เบื้องต้นสำหรับทั้งสองโพรโตคอล ซึ่งเราพบการโจมตีสองแบบในอีซีเอสวันและการโจมตีสามแบบในอีซีเอสวันที่ถูกปรับปรุงโดยเบาว์ แต่การวิเคราะห์ในงานที่แล้วของเรามีข้อจำกัดและไม่สมบูรณ์ ในบทความนี้เราได้ทำการขยายการวิเคราะห์ความมั่นคงปลอดภัยให้ละเอียดเพิ่มมากขึ้น ซึ่งทำให้เราค้นพบการโจมตีใหม่สองแบบในลักษณะมัลติเซสชั่นในอีซีเอสวันและการโจมตีใหม่สองแบบในลักษณะมัลติเซสชั่นในอีซีเอสวันที่ถูกปรับปรุงโดยเบาว์

คำสำคัญ: วิธีการแบบฟอร์มอลในการวิเคราะห์โพรโตคอลที่ใช้เทคนิคการเข้ารหัสข้อมูล วิธีการโมเดลเช็คกิ้ง วิธีการแบบฟอร์มอล คัลเลอร์เพทริเน็ต โพรโตคอลที่ใช้เทคนิคการเข้ารหัสข้อมูลเพื่อความปลอดภัย การรักษาความปลอดภัยในเครือข่าย

ห้องปฏิบัติการวิจัยตรรกศาสตร์และการรักษาความมั่นคงปลอดภัย ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี

*ผู้นิพนธ์ประสานงาน, e-mail yongyuth.per@kmutt.ac.th

An Improved Security Analysis of Micali's Fair Contract Signing Protocol by Using Coloured Petri Nets

Yongyuth Permpoontanalarp*

ABSTRACT

Micali proposed a simple and practical cryptographic fair exchange protocol, called ECS1, for contract signing. Bao et al. found some message replay attacks in both the original ECS1 and a modified ECS1 where the latter aims to solve an ambiguity in the former. Later, Zhang et. al. found two multi-session attacks in the modified ECS1. Both Bao's and Zhang's analyses are manual. In our previous work, we developed an automated methodology to analyze cryptographic protocols by using Coloured Petri Nets (CPN) and performed a preliminary analysis on both versions of the ECS1. There, we found two multi-session attacks in the original ECS1 and three multi-session attacks in the modified ECS1 but our previous analysis is restricted and incomplete. In this paper, we extended our previous analysis on the ECS1 more comprehensively. As a result, we found two new multi-session attacks in the original ECS1 and two new multi-session attacks in the modified ECS1.

Keywords: Formal methods for cryptographic protocols, model checking, coloured petri nets, cryptographic protocols, network security, formal methods

Logic and security lab, Department of Computer Engineering, Faculty of Engineering, King Mongkut's University of Technology Thonburi

*Corresponding author, email: yongyuth.per@kmutt.ac.th

Introduction

Cryptographic protocols are protocols which use cryptographic techniques to achieve certain tasks while preventing malicious parties to attack the protocols. There are many applications of cryptographic protocols, for example, authenticated key exchange protocols, web security protocols, e-payment protocols, e-banking protocols, e-voting protocols, etc.

The design and analysis of cryptographic protocols are difficult to achieve because of the increasingly attacking capabilities and the complex requirement of the applications. Attacks in many cryptographic protocols have been found later after the protocols have been designed and even implemented eg. [1-4]. Thus, it requires a method to analyze all possible attacks to the protocols. Such kind of method would offer a comprehensive understanding of all vulnerabilities of protocols and certainly would help in developing a better protection for them. Note that in this paper we focus on only message replay attacks [5]. Also, we focus on an analysis of multiple executions of a protocol which is also called multi-sessions of protocol execution.

In [6], Micali proposed a cryptographic fair exchange protocol, called ECS1, for contract signing. A fair exchange protocol ensures that either the two exchanging parties get the exchanged messages or none of them obtain anything. Fair contract signing protocol is a kind of fair exchange protocol where two parties aim to exchange their digital signature on an agreed contract. The optimistic contract signing protocol means that a trusted third party (TTP) is involved only when there is a dispute between the exchanging parties.

Bao et al. [7] analyzed ECS1 and found several single-session message replay attacks in ECS1. In general, they showed that the protocol does not achieve the claimed fairness. It can be argued that many of those attacks are caused by an ambiguity at how TTP should resolve a dispute. Then, Bao et al. showed that a modified version of ECS1 which aims to solve the ambiguity in the original ECS1 does not provide the fairness also. In particular, Bao et. al. found one multi-session attack to the modified ECS1. Later on, Zhang et al. [8] found two multi-session attacks in the modified ECS1. Those attacks lead to the unfair exchange. It should be noted that both Bao's and Zhang's analyses are manual.

Coloured Petri Net (CPN) [9,10] is a formal method which is based on graph theory. CPN offers an automated analysis and it has been applied to analyze distributed systems and communication protocols. In fact, CPN is a model checking technique where a system is first modeled by a kind of graphs, called a net, and then a state space of all possible executions of the system is generated and analyzed to search for errors in the system. CPN has many advantages in that it offers an intuitive and easy way to model a system and it provides many kinds of interesting analysis on the output state space. Furthermore, it has a software tool which facilitates the creation, the modification and the analysis of nets.

In our previous work [11,12], we have developed a methodology to analyze cryptographic protocols by using CPN. Our method can analyze multiple and concurrent sessions of protocol execution, and can detect multiple attack traces in the protocols. In [13,14], we applied our methodology to perform a preliminary analysis on ECS1 and found two multi-session attacks in original ECS1 and three multi-session attacks in the modified ECS1. There, our analysis is preliminary in that our attacker model for ECS1 is restricted and incomplete.

In this paper, we extend our previous attacker model more comprehensively for ECS1 and perform the analysis. We found two new multi-session attacks in original ECS1 and two new multi-session attacks in the modified ECS1.

In background section, we discuss Micali's original ECS1 and Bao's modified ECS1. Also, all previous attack analyses on both versions of ECS1 are discussed. In related-work section, we compare our CPN method to existing related works. In methodology section, our CPN methodology and CPN model for ECS1 are presented. In result section, our results on new attacks found by our CPN method are discussed.

Background

We use the following notations throughout the paper. $S \rightarrow R : M$ means that user S sends message M to user R. $SIG_X(M)$ represents party X's signature on a message M and we assume that plaintext M is retrievable from $SIG_X(M)$. The encryption of a message M with party X's public key is denoted by $E_X(M)$.

Alice and Bob stand for well-behaved initiator and responder, respectively. TTP means a trusted third party who facilitates the exchange between Alice and Bob. I denotes an attacker who can act as initiator or responder, and $I(S)$ means that the attacker impersonates user S in the protocol.

A single session means the single execution of a protocol whereas multi-sessions mean the multiple and concurrent executions of a protocol. While steps 1)-6) describe protocol steps in the first session, steps 1')-6') describe protocol steps in the second session.

Micali's ECS1 Protocol [6]

Micali proposed a cryptographic fair exchange protocol for contract signing. The protocol aims to ensure that two exchanging parties get each other commitment on an agreed contract or neither of them does. There are three kinds of parties in the protocol : Alice as an initiator of the protocol, Bob as an responder of the protocol and a third trusted party who resolves a dispute between Alice and Bob during the exchange.

We denote Alice, Bob and a trusted party by A, B and TTP, respectively. It is assumed that both Alice and Bob have already agreed on a plaintext contract C before the exchange. Alice is committed to contract C as an initiator if Bob has both $SIG_A(C,Z)$ and M where $Z = E_{TTP}(A,B,M)$ and M is a random message. On the other hand, Bob is committed to C as a responder if Alice has both $SIG_B(C,Z)$ and $SIG_B(Z)$. However, there is no need for Alice to verify *Z* to prove Bob's commitment.

The following is the detail of the protocol.

```
A1:  1) $A \rightarrow B$: $SIG_A(C,Z)$
B1:  2) $B \rightarrow A$: $SIG_B(C,Z)$, $SIG_B(Z)$
A2:  If Bob's signatures in step 2 are both valid then
     3) $A \rightarrow B$: M
B2:  If Bob receives valid M such that $Z = E_{TTP}(A,B,M)$
     then the exchange is completed
     else Bob requests TTP to resolve a dispute by the following step
     4) $B \rightarrow TTP$: $SIG_A(C,Z)$, $SIG_B(C,Z)$, $SIG_B(Z)$
TTP1: If Both Alice's and Bob's signatures in step 4 are valid and
     $Z = E_{TTP}(A,B,M)$ then
     5) $TTP \rightarrow A$: $SIG_B(C,Z)$, $SIG_B(Z)$
     6) $TTP \rightarrow B$: M
```

To resolve the dispute, TTP sends required information to related parties.

Bao et al.'s attack on ECS1 [7]

Bao et al. found two single-session message replay attacks in ECS1. It can be argued that many of these attacks are caused by an ambiguity at how TTP should resolve a dispute. In particular, at TTP1 if $Z = E_{TTP}(A,B,M)$ for some M, then TTP will resolve the dispute. Otherwise, the protocol does not specify what TTP should do. Then, Bao proposed a modified version of ECS1 which aims to solve the ambiguity in the original ECS1. The modified ECS1 is similar to the original ECS1 except that in step TTP1, there is no check on $Z = E_{TTP}(A,B,M)$. In other words, in the modified ECS1, TTP resolves the dispute even when Z is not correct, ie. $Z \neq E_{TTP}(A, B, M)$. Bao showed that the modified version of ECS1 can also be attacked as follows.

In the attack, malicious responder gains an advantage over well-behaved initiator Alice. In particular, malicious responder I obtains Alice's commitment, but Alice does not get I's commitment. There are two attackers. Malicious responder I conspires with another malicious initiator Ar in a session. In the first session, Alice exchanges with malicious responder I on contract C, and in the second session conspired initiator Ar exchanges with responder I on another contract C'.

1) $A \rightarrow I$: $SIG_A(C,Z)$ where $Z = E_{TTP}(A,I,M)$

2) $I \rightarrow A$: Nothing

Before the second session starts, I secretly sends Z to Ar.

1') $Ar \rightarrow I$: $SIG_{Ar}(C',Z)$

4') $I \rightarrow TTP$: $SIG_{Ar}(C',Z)$, $SIG_I(C',Z)$, $SIG_I(Z)$

5') $TTP \rightarrow Ar$: $SIG_I(C',Z)$, $SIG_I(Z)$

6') $TTP \rightarrow I$: M

Note that in the second session, malicious I skips protocol steps 2 and 3 and then requests TTP for dispute resolution. Also, in step 2) of the first session, I aborts the protocol by sending nothing.

Zhang et al.'s attack on ECS1 [8]

Zhang et al. analyzed both ECS1 and the modified ECS1 and they found one single session attack in the original ECS1 and two multi-session attacks in the modified ECS1. Since in this paper we focus on multi-session attacks, only Zhang's two multi-session attacks are described here. In both attacks, malicious responder gains an advantage over well-behaved initiator Alice.

In the first attack, malicious responder I obtains Alice's commitment, but Alice does not get I's commitment. In the first session, Alice exchanges with malicious responder I on contract C. In the second session, I as initiator exchanges with Bob on contract C'.

1) $A \rightarrow I$: $SIG_A(C,Z)$ where $Z = E_{TTP}(A,I,M)$

2) $I \rightarrow A$: Nothing

1') $I \rightarrow B$: $SIG_I(C',Z)$

2') $B \rightarrow I$: $SIG_B(C',Z)$, $SIG_B(Z)$

4') $I \rightarrow TTP$: SIG_{BI}, $SIG_I(C',Z)$, $SIG_I(Z)$

5') $TTP \rightarrow B$: $SIG_I(C',Z)$, $SIG_I(Z)$

6') $TTP \rightarrow I$: M

Note that in the second session, malicious I skips the protocol step 3 and then requests TTP for dispute resolution as a responder in the session. Note also that at step 5') of the second session, Bob does not obtain I's commitment since Z is for the session between Alice and I. In other words, Z is not valid for Bob. Note also that the first Zhang's attack achieves the same damage as Bao's attack but Zhang's attack requires only one attacker.

In the second attack, malicious responder I obtains Alice's commitment, but Alice does not get I's commitment. In the first session, Alice exchanges with malicious responder I. In the second session, Alice exchanges with Bob. In both sessions, the same contract is signed.

1) $A \rightarrow I : SIG_A(C,Z)$ where $Z = E_{TTP}(A,I,M)$
2) $I \rightarrow A$: Nothing
   1') $A \rightarrow I(B) : SIG_A(C,Z')$ where $Z' = E_{TTP}(A,B,M')$
   $I(A) \rightarrow B : SIG_A(C,Z)$
   2') $B \rightarrow I(A) : SIG_B(C,Z), SIG_B(Z)$
   4') $I \rightarrow TTP : SIG_B(C,Z), SIG_I(C,Z), SIG_I(Z)$
   5') $TTP \rightarrow B : SIG_I(C,Z), SIG_I(Z)$
   6') $TTP \rightarrow I : M$

Note that in step 1') of the second session, the message that is sent by Alice to Bob is intercepted and modified by I. In particular, I eavesdrops message sent by Alice to Bob by impersonating Bob. Then, I sends a modified message to Bob by impersonating Alice. It can be seen that the difference between Zhang's first attack and Zhang's second attack is on the contract in two sessions. While the second attack requires that the same contact is signed in two sessions, the first attack relaxes this requirement.

Our previous attacks on ECS1 [14]

In our previous work [11,12], we have developed a methodology to analyze cryptographic protocols by using CPN. In [13,14], we applied our methodology to perform a preliminary analysis on ECS1 and found two multi-session attacks in original ECS1 and three multi-session attacks in the modified ECS1. In the first attack on the original ECS1, malicious responder gains an advantage over well-behaved initiator Alice. In the second attack on the original ECS1, well-behaved initiator Alice gains an advantage over well-behaved responder Bob. For the modified ECS1, the first attack allows malicious responder to gain an advantage over well-behaved initiator Alice. In the second attack on modified ECS1, both well-behaved initiator Alice and malicious initiator I gain advantages over well-behaved responder Bob. In the third attack on modified ECS1, malicious responder I gains an advantage over Alice.

We discuss our previous attacks in the original ECS1 first. In the first attack, malicious responder I gets Alice's commitment, but Alice does not obtain I's commitment. In both sessions, Alice exchanges with malicious responder I by using the same random message M. But the contracts are different in the two sessions.

1) $A \rightarrow I : SIG_A(C,Z)$ where $Z = E_{TTP}(A,I,M)$
   1') $A \rightarrow I : SIG_A(C',Z)$
   2') $I \rightarrow A$: Nothing
2) $I \rightarrow A : SIG_I(C,Z), SIG_I(Z)$
3) $A \rightarrow I : M$

Since M is used in both sessions, at step 3) of the first session I also get Alice's commitment for the first session too.

In the second attack, Alice obtains Bob's commitment but Bob does not get Alice's commitment. In the first session, Alice exchanges with malicious responder I. In the second session, Alice exchanges with Bob. The same contract is signed in both sessions.

1) $A \rightarrow I : SIG_A(C,Z)$ where $Z = E_{TTP}(A,I,M)$

2) $I \rightarrow A$: Nothing

1') $A \rightarrow I(B) : SIG_A(C,Z')$ where $Z' = E_{TTP}(A,B,M')$

$I(A) \rightarrow B : SIG_A(C,Z)$

2') $B \rightarrow A : SIG_B(C,Z), SIG_B(Z)$

3') $A \rightarrow B : M'$

4') $B \rightarrow TTP : SIG_A(C,Z), SIG_B(C,Z), SIG_B(Z)$

5') $TTP \rightarrow A$: Error

6') $TTP \rightarrow B$: Error

At steps 5') and 6') of the second session, TTP does not send any information to Alice and Bob to resolve the dispute since Z is not valid. Also, the dispute resolution process is in an error state.

In the following, we discuss our three multi-session attacks on the modified ECS1. In the first attack, malicious responder I obtains Alice's commitment, but Alice does not get I's commitment. There are two attackers. Attacker I conspires with another malicious initiator Ar in a session. In the first session, Alice exchanges with malicious responder I, and in the second session conspired initiator Ar exchanges with malicious responder I.

1) $A \rightarrow I : SIG_A(C,Z)$ where $Z = E_{TTP}(A,I,M)$

2) $I \rightarrow A$: Nothing

Before the second session starts, I secretly sends $SIG_A(C,Z)$ to Ar.

4') $Ar \rightarrow TTP : SIG_A(C,Z), SIG_{Ar}(C,Z), SIG_{Ar}(Z)$

5') $TTP \rightarrow A : SIG_{Ar}(C,Z), SIG_{Ar}(Z)$

6') $TTP \rightarrow Ar : M$

Then, Ar sends secretly M to I.

Note that Alice obtains Ar's commitment instead at step 5') of the second session. This is not useful since Alice expects to get I's commitment. Note also that this attack looks similar to Bao's attack but the difference is on the party who requests for the dispute resolution to TTP. In Bao's attack, malicious attacker I sends a dispute resolution request, but in our attack the conspired party Ar sends the request.

In the second attack, both Alice and malicious initiator I obtain Bob's commitment, but Bob get neither Alice's and I's commitments. In the first session, Alice exchanges with Bob and in the second session, malicious initiator I exchanges with Bob. In both sessions, the same contract is signed.

1) $A \rightarrow I(B) : SIG_A(C,Z)$ where $Z = E_{TTP}(A,B,M)$
   1') $I \rightarrow B : SIG_I(C,Z)$
   2') $B \rightarrow I : SIG_B(C,Z), SIG_B(Z)$
   3') $I \rightarrow B$: Nothing
   4') $B \rightarrow TTP : SIG_I(C,Z), SIG_B(C,Z), SIG_B(Z)$
   5') $TTP \rightarrow I : SIG_B(C,Z), SIG_B(Z)$
   6') $TTP \rightarrow B : M$
2) $I(B) \rightarrow A : SIG_B(C,Z), SIG_B(Z)$
3) $A \rightarrow I(B) : M$

Note that in the first session, I impersonates B and all exchanged messages have never arrived to B.

In the third attack, malicious responder I obtains Alice's commitment, but Alice gets I's commitment on an unintended contract. In the first session, Alice exchanges with malicious responder I on contract C. In the second session, Alice exchanges with Bob on contract C'. In both sessions, the same random message M is used, but different contracts are signed.

1) $A \rightarrow I : SIG_A(C,Z)$ where $Z = E_{TTP}(A,I,M)$
   1') $A \rightarrow I(B) : SIG_A(C',Z')$ where $Z' = E_{TTP}(A,B,M)$
   2') $I(B) \rightarrow B$: Nothing
   4) $I \rightarrow TTP : SIG_A(C',Z'), SIG_I(C',Z'), SIG_I(Z')$
   5) $TTP \rightarrow A : SIG_I(C',Z'), SIG_I(Z')$
   6) $TTP \rightarrow I : M$

Note that at step 2'), the second session is aborted. Exchanged messages have never arrived to Bob. At the end of the first session, Alice obtains I's commitment on contract C', instead of C.

Coloured Petri Nets (CPN) [9, 10]

CPN is a formal method based on graph theory to analyze distributed systems and communication protocols. In fact, CPN is a model checking technique where a system is first modeled by a kind of graphs, called a net and then a state space of all possible executions of the system is generated and analyzed to search for errors in the system. However, only a single error trace can be detected by the built-in mechanism in CPN.

CPN extends Petri nets [15] with programming abilities. The programming language provided in CPN is a functional programming language called CPN-ML, and it is based on standard ML. A graph model in CPN contains four main components: places, transitions, arcs and tokens. Places, represented by circles, and transitions, represented by rectangles, are used to describe states and actions of the system, respectively. Arcs represented by arrows are used to link between place and transition. Tokens mean data and they are just a rich set of data types in standard ML. Arc expressions which are CPN-ML programs attached to arcs describe how a transition produces output tokens from input tokens. A transition can occur, called enabled, only when there are sufficient tokens that match the arc expressions on its input places. When an enabled transition is executed, called occurred, tokens from input places are removed and new tokens are added into output places according to corresponding arc expressions. Thus, the main programming functionality in CPN is to perform token processing at transitions. Furthermore, CPN provides a software tool called CPNTools [10] which facilitates the creation, the modification and the analysis of nets.

The original Petri nets [15], called place/transition nets, are considered as low-level nets. It is widely known that low-level nets are not practical to deal with real-world applications due to their unmanageable size of the net specification. Thus, many high-level nets, for example CPN and Predicate/transition nets [27], have been developed to solve this problem. In fact, CPN is very much similar to Predicate/transition nets [27] and they are considered as two different dialects. There are other kinds of high-level nets, for example algebraic Petri nets [28], CPN with algebraic specifications [29], etc. Most of them are similar to CPN, but the difference is on the inscription language which is the language for arc expressions and tokens. While the inscription language in CPN is based on functional programming, the inscription language in algebraic Petri nets and CPN with algebraic specifications is based on algebraic specifications. Further discussion on the difference between CPN and other high-level nets can be found in [9].

Related works

Petri Nets for Cryptographic Protocols

A lot of works on Petri nets (PN) [16-23] have been applied to analyze cryptographic protocols. They can be classified into two kinds. The first kind [16-22] offers a modeling and analysis method to find attacks. The second kind [23] provides a theoretical semantics for cryptographic protocols which can be used to prove properties of protocols, rather than to find attacks. We focus on the former kind due to its practical use. All the works in the first kind except for [22] offer an analysis of a single session of protocol execution only, but all of them can analyze a single attack trace only.

The works in [16, 17] are the first which applies Petri nets to analyze cryptographic protocols. In particular, they employ CPN to detect attacks in the protocols. However, they analyze strictly a single session of protocol execution and can detect a single attack trace only. Later on, the works in [18,19] developed place/transition nets which are low-level nets to analyze security protocols. However, those low-level nets are less expressive than CPN which is considered as high-level nets. In particular, tokens in those low-level nets are uncoloured or just dots which represent an availability status of some resource or data only whereas tokens in CPN are rich data types in standard ML. Furthermore, CPN offers the programming ability to perform token processing at transitions, but this ability is missing in those low-level nets.

Recently, there are two works [20, 21] which apply CPN to analyze cryptographic protocols. Both works do not truly provide an analysis of multiple concurrent sessions of protocol execution. Even though [20] offers an analysis of two sequential sessions of protocol execution, it has to analyze one session at a time. So, it is considered as a single session analysis. Thus, the attacker model in both works has limited capabilities. The work [22] offers an analysis of multiple and concurrent session of protocol execution. However, like all other Petri-net approaches [16-21], the work can detect a single attack trace only. But our CPN method not only can analyze multiple and concurrent sessions of protocol execution but also can detect multiple attack traces in the protocol.

Other formal methods for analyzing Contract Signing protocols

In [24], Chadha, Kanovich, and Scedrov proposed an inductive proof method to analyze a variant of contract-signing protocol proposed by Garay, Jakobsson and MacKenzie. Their method aims to prove fairness and abuse-free properties of the protocol, rather than to find attacks. In addition, their method is manual. In [25], Shamatikov and Mitchell applied a model checking system called Murφ to analyze two contract signing protocols. A protocol is modeled as an automata by using a programming language. Their method is automatic and some new attacks on the protocols are discovered. In [26], Gurgens and Rudolph analyzed a number of fair exchange non-repudiation protocols using asynchronous product automata (APA) and the simple homomorphism verification tool (SHVT). Similar to Murφ, a protocol is modeled as an automata but by using a text-based description on states and state transitions. Their method is automatic and it found new attacks. Both Murφ and SHVT offer an analysis of multiple sessions, but they can detect only a single attack trace.

Our Model

Our Methodology [11, 12]

In general, our model checking methodology for the analysis of cryptographic protocols consists of five steps which are (1) protocol and attacker representation, (2) state space and trace generation, (3) characterization and search for attack states, (4) attack trace extraction and (5) attack trace classification. The detail of our methodology is shown as follows.

First, we build a CPN graph model to represent message exchange by all user parties and to represent attacker behavior. Each user party is modeled according to the protocol. Moreover, an attacker in our model can eavesdrop, modify and drop messages during the transmission. Also, the attacker can send new messages. Both user and attacker representations are present in the same model.

Second, an automata or a state space of the protocol with attackers is generated by using the state space tool in CPNTools [10]. In general, the state space represents all possible behaviors of every party, including attacker, in the protocol. To reduce the size of a computed state space in our method, we employ a decomposition technique. In particular, we define a configuration to compute a decomposed state space. In this paper, we consider the analysis of multi-sessions of protocol execution. A configuration consists of the information for the protocol execution in a multi-session setting, for example, the identities of initiator and responder, the role of attackers, secrets and nounces in each concurrent session, and a schedule of the execution of the multiple concurrent sessions. The schedule specifies that the decomposed state space is computed for one alternating execution of multiple concurrent sessions of protocol runs only, instead of all possible alternating executions. Exploring all possible alternating executions within a state space is expensive and causes a huge state space. However, we can explore each attack scenario, eg. a specific alternating execution or a specific initiator and responder, one by one by computing a decomposed state space with a specific configuration.

Third, we create a query function in CPNML language, which is based on ML functional programming language, to search for attack states in the state space. Attack states are characterized by vulnerability events. Vulnerability events are events which may lead to a compromise of protocols, and such events are protocol dependent. In ECS1, there is one vulnerability event where one party, who is either initiator or responder, gets another party commitment, but the latter does not get the former commitment. In other words, this event describes exactly an unfair state. The concept of vulnerability events provides a general method to characterize attack states intuitively and comprehensively. Also, queries can be built easily to detect such combined vulnerability events.

Fourth, after attack states are discovered from the state space, we extract attack traces. Conceptually, an attack trace describes how an attacker carries out an attack successfully step by step. Given an attack state in a state space, a path from the initial state to the attack state is an attack trace. Our method offers an efficient new approach to extract attack traces from an output state space without the need for any further computation. In our CPN model, as the protocol execution proceeds, an attack trace which is a record of all exchanged messages between parties so far is stored into an output state. More specifically, the attack trace is stored in a global fusion place when a message is sent from one user to another. Thus, when an attack state is found, the attack trace can be extracted from the state immediately. As a result, the analysis of multiple attack traces of an attack can be performed efficiently by simple searching for attack states in the state space and extracting an attack trace from the attack state.

Fifth, after attack traces are obtained, we classify them into each group. In general, there can be a huge amount of attack states and traces found in a state space. For example, in ECS1 we found 22,890 attack traces in a configuration as shown in table 1. Thus, to ease the analysis of a large amount of attack traces, we develop an attack classification by using attack patterns. In general, an attack pattern describes the core of an attack, and it contains a list of minimal protocol messages that are the cause of each attack. Attack traces that produce the same attack pattern are classified into the same group of attacks. In general, it requires human intervention to create each attack pattern from an attack trace.

Our Extended Attacker Model

The following shows the description of our attacker model.

Definition 1: The attacker abilities

The attacker in our model is capable of the following:

1. The attacker can eavesdrop, modify and drop messages during the transmission between users.

2. The attacker can send a message to a user.

3. The attacker can either initiate a new session with users or take part in an existing session with users.

4. The attacker can impersonate any user.

5. The attacker can perform any cryptographic computation by using known keys, known messages and known ciphertexts with a limited but reasonable power, eg. encryption and decryption.

6. The attacker has its own storage with a finite and reasonable amount.

7. The attacker does not attack himself.

In our previous analysis [14], our attacker model is restricted and incomplete in many aspects. Firstly, the attacking ability to modify messages during the transmission does not apply to a message at protocol step 4 in both versions of ECS1. In other words, our previous attacker does not modify any dispute resolution request to TTP at all. Secondly, only an attacker who acts as a malicious responder in an exchange can send a dispute resolution request to TTP. So, an attacker who acts as a malicious initiator is not allowed to request for a dispute resolution to TTP. Thirdly, our attacker possesses only one random message M, instead of two random messages. This is restricted since it does not allow an analysis of attack scenarios where an attacker involve in two concurrent sessions as initiator, and the attacker uses two different random messages in the two sessions.

In our current analysis, we extend our previous attacker model to overcome all these restrictions. As a result, we obtain a more comprehensive attack analysis and find new attacks in both versions of ECS1.

Our CPN Model

Our CPN model consists of three levels: top, entity, and process levels. Due to space limit, we will present very few parts of our model only. The top level shows the interaction and communication between all parties. Figure 1 shows the top level net. It can be seen that all exchanged messages amongst A, B and TTP have to pass through attacker I, and I can conspire with Ar.

The entity level shows all actions that each party involves in the protocol. Each action or process corresponds to a protocol step. Alice's entity level consists of three processes: TA1, TA2, and TA3, and Bob's entity level also has three processes: TB1, TB2, and TB3. But Intruder's entity level contains six processes and so is Ar. There is no TTP's entity level for simplicity since TTP performs only one step. Alice's entity level is shown in figure 2.

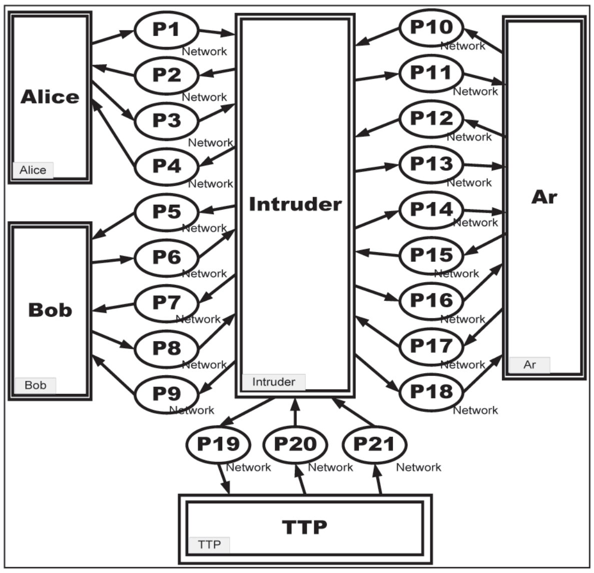

Figure 1 Top level

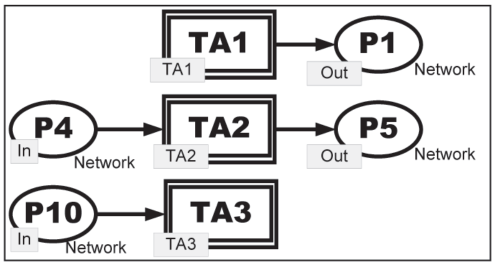

Figure 2 Alice's entity level

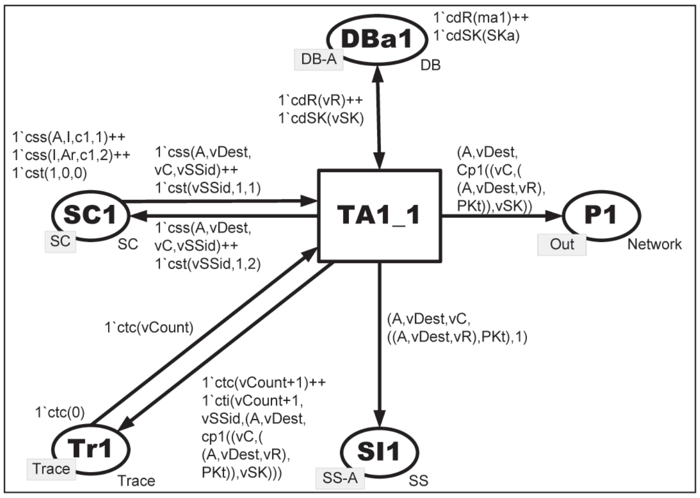

Figure 3 TA1's process level

The process level shows the detail of each process in the entity level. Figure 3 shows the detail of A's TA1 process which is to generate the message at step 1. The transition *TA1_1* is to generate A's signature on contract and Z by using her private key from her own DB-place and to send it to responder. The *SC1* place is to check session scheduling for A. If A's model is scheduled to execute, then an appropriate token will appear at *SC1* and then the transition *TA1_1* will occur. Also, an attack trace is stored into a state by recording the trace in the global fusion place *TR1*.

In addition, there is a session scheduling process which is to schedule each session to execute and to control the number of protocol steps to be executed in a scheduled session according to a given configuration. To simplify explanation, we omit the detail of the scheduling process here.

Result and Discussion

New Attacks in the original ECS1

We found two new attacks in the original ECS1 protocol. In the first attack, a malicious initiator and Alice gains an advantage over Bob. In the second attack, Alice gains an advantage over Bob on an unintended contract.

In the first attack, both Alice and malicious initiator I obtain Bob's commitment, but Bob get neither Alice's nor I's commitments. In the first session, Alice exchanges with Bob. In the second session, malicious initiator I exchanges with Bob. In both sessions, the same contract is signed.

1) $A \rightarrow I(B) : SIG_A(C,Z)$ where $Z = E_{TTP}(A,B,M)$
$I(A) \rightarrow B$: Nothing
1') $I \rightarrow B : SIG_I(C,Z)$
2') $B \rightarrow I : SIG_B(C,Z), SIG_B(Z)$
3') $I \rightarrow B : M'$ where $M' \neq M$
4') $B \rightarrow I(TTP) : SIG_I(C,Z), SIG_B(C,Z), SIG_B(Z)$
$I(B) \rightarrow TTP : SIG_A(C,Z), SIG_B(C,Z), SIG_B(Z)$
5') $TTP \rightarrow A : SIG_B(C,Z), SIG_B(Z)$
6') $TTP \rightarrow B : M$

In step 1) of the first session, the message is intercepted by attacker I and is dropped. In other words, it has never arrived to Bob. In step 4') of the second session, the dispute resolution request is intercepted by attacker I, and I sends a modified request to TTP. It can be seen that at the end of the dispute resolution, TTP is fooled to forward Bob's commitment to Alice.

In fact, our new first attack achieves the same damage as the second attack in our previous analysis, but the attack methods are different. In our previous analysis, it was attacker I who sends Bob's commitment to Alice. But in our new attack, it is TTP who sends Bob's commitment. We consider our new attack as a more reasonable one since TTP is fooled to take part in the attack. But in the previous analysis, attacker has to explicitly send Bob's commitment to Alice.

In the second attack, Alice obtains Bob's commitment on an unintended contract, but Bob does not get Alice's commitment. In the first session, Alice exchanges with Bob on contract C. In the second session, malicious initiator I exchanges with Bob on another contract C'. At the end of the first session, Alice obtains Bob's commitment on contract C' instead of C.

1) $A \rightarrow B : SIG_A(C,Z)$ where $Z = E_{TTP}(A,B,M)$

   1') $I \rightarrow B : SIG_I(C',Z')$ where $Z = E_{TTP}(I,B,M')$

   2') $B \rightarrow I : SIG_B(C',Z'), SIG_B(Z')$

   3') $I \rightarrow B : M'$

2) $B \rightarrow I(A) : SIG_B(C,Z), SIG_B(Z)$

   $I(B) \rightarrow A : SIG_B(C',Z'), SIG_B(Z')$

3) $A \rightarrow B$: Nothing

4) $B \rightarrow I(TTP) : SIG_A(C,Z), SIG_B(C,Z), SIG_B(Z)$

   $I(B) \rightarrow TTP : SIG_I(C',Z'), SIG_B(C',Z'), SIG_B(Z')$

5) $TTP \rightarrow I : SIG_B(C',Z'), SIG_B(Z')$

6) $TTP \rightarrow B : M'$

At step 2) of the first session, the message is replayed from the second session. So, Alice obtains Bob's commitment on contract C'. At step 4) of the first session, the dispute resolution request is intercepted by attacker I, and the attacker sends a modified request by using messages from the second session. Indeed, the dispute resolution request at step 4) of the first session can be dropped by the attacker, and the result of the attack is still the same.

New Attacks in Bao's modified ECS1

We found two new attacks in Bao's modified ECS1 protocol. In the first attack, Bob gains an advantage over Alice, and malicious initiator gains an advantage over Bob. In the second attack, Alice gains an advantage over Bob on an unintended contract.

In the first attack, Bob obtains Alice's commitment, but Alice gets Bob's commitment on an unintended contract. Moreover, malicious initiator I obtains Bob's commitment, but Bob does not have I's commitment. In the first session, Alice exchanges with Bob on a contract. In the second session, I exchanges with Bob on a different contract.

1) $A \rightarrow B$: SIG_A(C,Z) where Z = E_{TTP}(A,B,M)
   1') $I \rightarrow B$: SIG_I(C',Z)
   2') $B \rightarrow I$: SIG_B(C',Z), SIG_B(Z)
   3') $I \rightarrow B$: M'
   4') $B \rightarrow$ I(TTP) : SIG_I(C',Z), SIG_B(C',Z), SIG_B(Z)
   5') I(TTP) $\rightarrow$ TTP : Nothing
2) $B \rightarrow$ I(A) : SIG_B(C,Z), SIG_B(Z)
   I(B) $\rightarrow A$: SIG_I(C',Z), SIG_B(Z)
3) $A \rightarrow B$: Nothing
4) $B \rightarrow$ I(TTP) : SIG_A(C,Z), SIG_B(C,Z), SIG_B(Z)
   I(B) $\rightarrow$ TTP : SIG_I(C',Z), SIG_B(C',Z), SIG_B(Z)
5) TTP $\rightarrow$ I : SIG_B(C',Z), SIG_B(Z)
6) TTP $\rightarrow$ B : M

Note that at step 4) of the second session, malicious attacker intercepts the dispute resolution request from B and drops it off. So the request does not arrive to TTP. At step 2) of the first session, Bob's commitment on contract C is eavesdropped by attacker and Bob's commitment on another contract C' is replayed instead. At step 4) of the first session, the dispute resolution request where A is the initiator is intercepted and another request where I is the initiator is sent instead. As a result, Alice does not get Bob's commitment on contract C, But Bob obtains Alice's commitment.

Indeed, we can obtain another variant attack where Alice does not obtain Bob's commitment on an unintended contract, but Bob gets Alice's commitment and malicious initiator I obtains Bob's commitment. The variant attack is similar to this attack, except for step 2) of the first session where nothing is sent to Alice. In other words, Bob's commitment is simply dropped off by attacker.

It can be seen that in Bao's attack, Zhang's two attacks and the first and third attacks in our previous analysis, malicious responder gains an advantage over Alice also. In all of those attacks, the attacker tries to create a modified dispute resolution request to TTP where the responder in the request is the attacker. As a result of the resolution, the attacker would obtain the random message M and then Alice's commitment. However, our new attack uses a different approach in that the attacker tries to create a modified resolution request to TTP where the responder is Bob but the initiator is not Alice. So, Bob will obtain Alice's commitment, but Alice does not. In other words, Bob gains an advantage over Alice. In summary, our new attack allows the well-behaved responder, instead of attacker, to gain an advantage over Alice.

The second new attack is identical to our second new attack in the original ECS1 that is discussed previously. In other words, our second new attack works for both original ECS1 and the modified ECS1.

Performance

The table 1 shows some examples of configurations and their computation results. Each row of the table shows a simplified form of a configuration of a session. For example, *(A,I,c1,ma1)* means that in the session *A* and *I* are initiator and responder, respectively, and *c1* is a contract and *ma1* is the random message M. "Attack traces" means the number of attack traces found in the state space. "St" means the amount of times used to generate a state space, and "Tr" means the amount of times used to compute all attack traces. For the largest state space, the number of nodes is 564,098, and the computation times for state space and all attack traces are about 8 hours and 50 minutes, and about 4 minutes, respectively. For the smallest state space, the number of nodes is 1,225, and the computation times for state space and all attack traces are just 11 seconds and less than 1 second, respectively.

Table 1 Some results of the state space computation

Configurations of two sessions	**Attack Traces**	**Size of State Space**		**Times (in seconds)**		
		Nodes	**Arcs**	**St**	**Tr**	**Total**
1. (I,B,c1,mi1)(I,Ar,c2,mi2)	22,890	564,098	570,997	31,785	226	32,011
2. (I,B,c1,mi1)(I,Ar,c2,mi1)	23,652	522,680	530,246	28,879	240	29,119
3. (I,Ar,c2,mi1)(I,B,c1,mi2)	20,232	471,884	472,895	23,490	178	23,668
4. (I,Ar,c2,mi1)(I,B,c1,mi1)	18,726	436,151	437,141	20,634	154	20,788
5. (I,B,c1,mi1)(Ar,I,c2,mi1)	14,252	254,652	257,745	7,711	84	7,795
6. (I,B,c1,mi1)(Ar,I,c2,mi2)	12,884	233,210	235,509	6,478	71	6,549
7. (I,B,c1,mi1)(I,B,c2,mi2)	7,987	176,536	184,171	3,828	50	3,878
8. (A,I,c1,ma1)(I,Ar,c2,mi1)	1,254	129,372	129,371	3,264	7	3,271
9. (A,B,c1,ma1)(I,B,c1,mi1)	5,406	67,042	67,409	1,517	13	1,530
10. (A,B,c1,ma1)(I,B,c2,mi1)	3,935	48,303	48,590	989	7	1,006
11. (A,I,c1,ma1)(A,B,c1,ma1)	392	7,009	7,053	105	0	105
12. (A,B,c1,ma1)(A,I,c1,ma1)	108	3,375	3,386	46	0	46
13. (A,I,c1,ma1)(A,I,c2,ma1)	28	1,225	1,224	11	0	11

Conclusion

In this paper, we have extended our previous analysis on ECS1 by using our CPN-based methodology to analyze cryptographic protocols. In particular, we have extended our attacker model for both versions of ECS1 to allow a more comprehensive analysis than our previous attacker model. As a result, we found two new attacks in the original ECS1 and two attacks in the modified ECS1. This demonstrates clearly the usefulness of formal methods to analyze cryptographic protocols. Our CPN method allows an efficient way to analyze multiple attack traces in multiple sessions of protocol execution. As a future work, we aim to apply our method to analyze other kinds of cryptographic protocols.

Acknowledgement

The author would like to acknowledge a financial support from National Research Council of Thailand.

References

1. Clark, J., and Jacob, J. 1997. A Survey on Authentication Protocols. Available from URL: http://www.cs.york.ac.uk/~jac/papers/drareview.ps.gz. 15 October 2008.
2. Meadows, C. 2003. Formal Methods for Cryptographic Protocol Analysis: Emerging Issues and Trends. *IEEE Journal on Selected Areas in Communications* 21(1): 44-54.
3. Meyer, U., and Wetzel, S. 2004. A Man-In-The-Middle Attack on UMTS. In: Jakobsson, M. and Perrig, A. Editors. Proceedings of the 3rd ACM workshop on Wireless security, 1 October 2004. Philadelphia, U.S.A. ACM Press. p. 90-97
4. Cervesato, I., Jaggard, A. D., Scedrov, A., Tsay, J., and Walstad, C. 2008. Breaking and Fixing Public-Key Kerberos. *Information and Computation* 206(2-4): 402-424
5. Syverson, P. F. 1994. A Taxonomy of Replay Attacks. In: Proceedings of the 7th IEEE Computer Security Foundations Workshop, 14-16 June 1994. New Hampshire, U.S.A. IEEE Press. p. 187-191
6. Micali, S., 2003. Simple and Fast Optimistic Protocols for Fair Electronics Exchange. In: Proceedings of the 22nd Symposium on Principles of Distributed Computing. 13-16 July 2003. Boston, U.S.A. ACM press. p. 12-19
7. Bao, F., Wang, G., Zhou, J., and Zhu, Z. 2004. Analysis and Improvement of Micali's Fair Contract Signing Protocol. In: Wang, H., Pieprzyk, J., and Varadharajan, V., Editors. Proceedings of the 9th Australasian Conference on Information Security and Privacy, 13-15 July 2004, Sydney, Australia. LNCS Vol. 3108 Springer Verlag. p. 176-187

8. Zhang, Y., Wang, Z., and Yang, B. 2005. The Running-Mode Analysis of Two-Party Optimistic Fair Exchange Protocols. In: Hao Y, et. al., Editors. Proceedings of the International Conference on Computational Intelligence and Security. 15-19 December 2005. China. LNCS Vol. 3802. Springer Verlag. p. 137-142
9. Jensen, K. 1997. Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use, Vol.1. Monographs in Theoretical Computer Science. Berlin/Heidelberg. Springer-Verlag.
10. Jensen, K., Kristensen, L.M., and Wells, L. 2007. Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems. *International Journal on Software Tools for Technology Transfer* 9(3): 213-254.
11. Permpoontanalarp, Y., and Sornkhom, P. 2009. A New Coloured Petri Net Methodology for the Security Analysis of Cryptographic Protocols. In: Jensen, K., Editors. Proceedings of the 10th Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools. 19-21 October 2009. Aarhus. Denmark. p. 81-100.
12. Permpoontanalarp, Y. 2010. On-the-fly Trace Generation and Textual Trace Analysis and Their Applications to the Analysis of Cryptographic Protocols. In: Hatcliff, J., and Zucca, E., Editors. Proceedings of the 30th International Conference on Formal Techniques for Networked and Distributed Systems. 7-9 June 2010. Amsterdam, Netherlands. LNCS vol 6117 Springer-Verlag. p. 201-215.
13. Sornkhom, P., and Permpoontanalarp, Y. 2008. Security Analysis of Micali's Fair Contract Signing Protocol by Using Coloured Petri Nets. In: Proceedings of the 9th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/ Distributed Computing ACIS-SNPD. 6-8 August 2008, Phuket, Thailand. IEEE Press p. 329-334.
14. Sornkhom, P., and Permpoontanalarp, Y. 2009. Security Analysis of Micali's Fair Contract Signing Protocol by Using Coloured Petri Nets: Multi-Session Case. In: Proceedings of 23rd IEEE International Symposium on Parallel and Distributed Processing (the 5th International Workshop on Security in Systems and Networks). 23-29 May 2009. Rome, Italy. IEEE Press p. 1-8.
15. Murata, T. 1989. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE* 77(4): 541-580.
16. Nieh, B., and Tavares, S. 1992. Modelling and Analyzing Cryptographic Protocols Using Petri Nets, In: Seberry, J., and Zheng, Y., Editors. Advances in Cryptology-AUSCRYPT-92, Workshop on the Theory and Application of Cryptographic Techniques. 13-16 December 1992. Queensland, Australia. LNCS Volume 718. Springer Verlag, p. 275-295.

17. Basyouni, A., and Tavares, S. 1997. New Approach to Cryptographic Protocol Analysis using Coloured Petri Nets, In: Proceedings of the Canadian Conference on Electrical and Computer Engineering. 25-28 May 1997. Canada. IEEE Press. p. 334-337.
18. Lee, G., and Lee, J. 1997. Petri Net Based Models for Specification and Analysis of Cryptographic Protocols. *The Journal of Systems and Software* 37: 141-159.
19. Lim, S., Ko, J., Jun, E., and Lee, G. 2001. Specification and Analysis of N-Way Key Recovery System by Extended Cryptographic Timed Petri Net. *The Journal of Systems and Software* 58: 93-106.
20. Dresp, W. 2005. Security Analysis of the Secure Authentication Protocol by Means of Coloured Petri Nets. In: Dittmann, J., Katzenbeisser, S. and Uhl, A., Editors. Proceeding of the 9th IFIP Communications and Multimedia Security, 19-21 September 2005. Salzburg, Austria. LNCS, Springer Verlag. p. 230-239.
21. Al-Azzoni, I., Down, D. G., and Khedri, R. 2005. Modeling and Verification of Cryptographic Protocols Using Coloured Petri Nets and Design/CPN. *Nordic Journal of Computing* 12(3): 201-228.
22. Bouroulet, R., Devillers, R., Klaudel, H., Pelz, E., and Pommereau, F. 2008. Modeling and Analysis of Security Protocols Using Role Based Specifications and Petri Nets: In: Hee, K. M., and Valk, R., Editors. Proceeding of the 29th International Conference on Applications and Theory of Petri Nets. 23-27 June 2008, Xian, China. LNCS Vol. 5062 Springer Verlag, p. 72-91
23. Crazzolara, F., and Winskel, G. 2001. Events in Security Protocols, In: Proceedings of the 8th ACM Conference on Computer and Communication Security. 6-8 November 2001. Pennsylvania, U.S.A. ACM Press. p. 96-105.
24. Chadha, R., Konovich, M., and Scedrov, A. 2001. Inductive Methods and Contract-Signing Protocols. In: Proceedings of 8th ACM Conference on Computer and Communications Security. 6-8 November 2001. Pennsylvania, USA. ACM Press. p. 176-185.
25. Shmatikov, V., and Mitchell J.C. 2002. Finite-State Analysis of Two Contract Signing Protocols. *Theoretical Computer Science* 283: 419-450
26. Gürgens, S., and Rudolph, C. 2005. Security Analysis of Efficient (Un-) Fair Non-Repudiation Protocols. *Formal Aspect of Computing* 17(3): 260-276
27. Genrich, H. J., and Lautenbach, K. 1981. System Modelling with High-Level Petri Nets. *Theoretical Computer Science* 13(1): 109-135
28. Dimitrovici, C., Hummert, U., and Petrucci, L. 1991. Semantics, Composition and Net Properties of Algebraic High-level Nets. In: Rozenberg, G., Editors. Advances in Petri Nets. LNCS Vol. 524. Berlin/Heidelberg. Springer-Verlag. p. 93-117.

29. Vautherin, J. 1987. Parallel Systems Specifications with Colored Petri Nets and Algebraic Specifications. In: Rozenberg, G., Editors. Advances in Petri Nets. LNCS Vol. 266. Berlin/ Heidelberg. Springer-Verlag. p. 293-308.

ได้รับบทความวันที่ 5 กรกฎาคม 2553
ยอมรับตีพิมพ์วันที่ 18 สิงหาคม 2553