Modeling and Analysis of Security Protocols Using Colored Petri Nets

Yang Xu
Key Laboratory of Information and Computing Science of Guizhou Province, Guizhou Normal University, School of Computer Science and Technology, Guizhou University, Guiyang, China
Email: kyxy3465@sina.com.cn

Xiaoyao Xie
Key Laboratory of Information and Computing Science of Guizhou Province, Guizhou Normal University, Guiyang, China
Email: xyx@gznu.edu.cn

***Abstract*—Security protocols are the basis of security in networks. Therefore, it is essential to ensure that these protocols function correctly. However, it is difficult to design security protocols that are immune to malicious attack, since good analysis techniques are lacking. In this paper, the current main analysis techniques using Colored Petri Nets (CP-Nets) for analysis of security protocols are introduced. Based on the techniques, a new method using CP-Nets for the analysis of security protocols is presented. Specially, in the new method, an intruder CP-Net model is presented that provides an open-ended base for the integration of multiple attack tactics. This is a viable approach to overcome the state space explosion problem. Furthermore, the automated analysis tools CPN Tools is used. The Andrew secure RPC protocol is chosen to illustrate how a security protocol is analyzed using the new method. After model checking, an attack is found which the same as the one found by Gavin Lowe. These are stunning confirmations of the validity of the new method for analyzing security protocols.**

***Index Terms*—security protocols, protocol analysis, Colored Petri Net, CPN Tools, Andrew secure RPC protocol**

I. INTRODUCTION

With the rapid growth of security applications, network security has become an important issue, and security protocols are the basis of security in networks. Therefore, it is essential to ensure these protocols correctly. Unfortunately, it is difficult to design a robustness and effective security protocol for networks. Not only because of the characteristics of networks, but also because good analysis techniques are lacking.

In this paper, we compare the current main techniques using Colored Petri Nets (CP-Nets) [1] to analyze the security protocols. Based on the techniques, we present a new and promising method using CP-Nets for the analysis of security protocols. We adopt the assumptions of the Dolev-Yao model [2] and present a CP-Net intruder model that provides an open-ended base for the integration of multiple attack tactics. This is a viable approach to overcome the state space explosion problem. Furthermore, we show how to use the automated analysis tools CPN Tools [3] in the construction and model checking of the net models. Finally, we use the new method to model and analysis the Andrew secure RPC protocol fixed in [4] which uses the symmetric keys. In the example, we first introduce a CP-Net for the Andrew secure RPC protocol. Then, an intruder model is developed and integrated into the protocol model. Model checking is performed at last. In the model checking, two methods are used. One exploits the provided state space exploration functions and another is simulation implementation. After model checking and the state space analysis, an attack is found which the same as the one found by Gavin Lowe [5].

The paper is organized as follows. Section 2 is an overview of the main analysis methods currently using Petri Nets for analysis of security protocols. And a new method using CP-Nets to analyze security protocols is presented. Section 3 introduces the Andrew secure RPC protocol and its fixes [4]-[7]. In Section 4, a CP-Net for the Andrew secure RPC protocol fixed in [4] is introduced. Then, an intruder model is developed and integrated into the protocol model. In Section 5, model checking is performed in CPN Tools. After model checking and the state space analysis, an attack is found. Finally, we conclude the work and suggest future research in Section 6.

II. SECURITY PROTOCOLS ANALYSIS METHOD USING COLORED PETRI NETS

In this section, the definitions of Petri Nets and Colored Petri Nets are presented first. Then, the current main techniques using CP-Nets to analyze the security protocols are compared. Based on the techniques, we present a new method using CP-Nets to analyze the security protocols.

doi:10.4304/jcp.6.1.19-27

A. Petri Nets

Petri Nets are presented by Carl Adam Petri during his Ph.D. thesis in 1962. A Petri Net is a graphical and mathematical tool to verify systems and protocols. Petri Nets in the graphical forms are like flowcharts and network diagrams, while in mathematical forms, they are like algebra and logic subjects.

Definition 1:

In a formal way, A *Petri Net* is a tuple [8]:

$PN = (P, T, A, N)$

In the tuple,

(1) P is a finite set of *Places*.
(2) T is a finite set of *Transitions*.
(3) A is a finite set of *Arcs* such that:
$P \cap T = P \cap A = T \cap A = \varnothing$
(4) N is a set of *Token*.

B. Colored Petri Nets

There are two forms of Petri Nets: ordinary Petri Nets and high level Petri Nets. CP-Nets which belong to the high level Petri Nets.

Definition 2:

Within set theory, a *Multi-set* of S can be defined as a function from S to R where S is a non-empty set and R is a non-positive integers set [9].

Multi-set M:

$$\sum_{s \in S} m(s)`S$$

S_{MS} can be defined as the set consisting of all the multi-sets of S. The non-negative real numbers $\{m(s) | s \in S\}$ are the coefficients of multiple sets, and $S \in M$ *iff* $m(s) \neq 0$.

Definition 3:

In a formal way, A *CP-Net* is a tuple [1, 8]:

$CPN = (\sum, P, T, A, N, C, G, E, I)$

In the tuple,

(1) $\sum$ is a finite set of non-empty types, also called colored sets.
(2) P is a finite set of *Places*.
(3) T is a finite set of *Transitions*.
(4) A is a finite set of *Arcs* such that:
$P \cap T = P \cap A = T \cap A = \varnothing$.
(5) N is a node function. It is defined from A into "colored over arcs" $P \times T \cup T \cap P$.
(6) C is a color function. It is defined from P into $\sum$ "token".
(7) G is a guard function. It is defined from T into expressions such that: "Boolean function with probability."
$\forall t \in T : [Type(G(t)) = B \wedge Type(Var(G(t))) \subseteq \sum]$.
(8) E is an arc expression function. It is defined from A into expressions such that: i.e. (check k=n)
$\forall a \in A : [Type(E(a)) = C(p)_{MS} \wedge Type(Var(E(a))) \subseteq \sum]$
where P is the place of $N(a)$.
(9) I is an initialization function. It is defined from P into closed expressions such that
$\forall p \in P : [Type(I(p)) = C(p)_{MS}]$.

C. Protocol Analysis Using Colored Petri Nets

CP-Nets are suitable as a modeling technique to analyze and verify systems in different areas of science such as artificial intelligence, parallel processing system, control systems, and numerical analysis [8], [10]-[15]. CP-Nets have already proven also suitable as a modeling technique for analysis of security protocols [16]-[23]. They follow an elaborated mathematical syntax and provide a clear, intuitive and demonstrative graphical representation of the model thus facilitating its simulation and analysis which is a basic strength compared to other verification methods.

There are two courses for using CP-Nets: forward or backward analysis in the traditional techniques. In [19]-[22], the backward state analysis of a security protocol includes three steps:

1. Generating an explicit CP-Nets specification for the protocol.
2. Identifying insecure states that may or may not occur.
3. Performing a backward state analysis to test if each insecure state is reachable or not.

Unfortunately, the methods have the problems below:

1. Generally, a protocol model is applicable only to one specific attack, and can only describe one possible insecure state (Fig. 1).

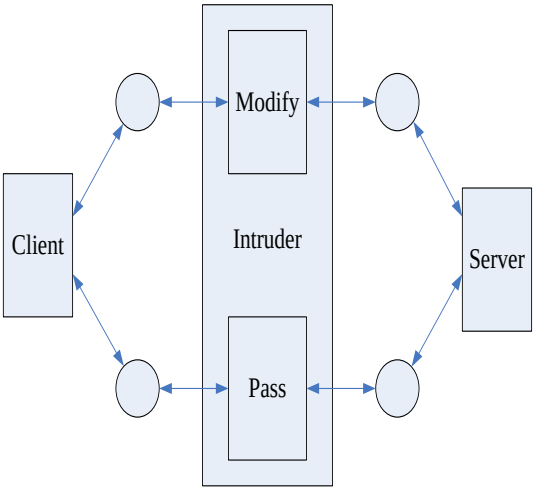

Figure 1. The traditional CP-Nets intruder model.

2. Automated analysis tools of CP-Nets are not be used. Thus, the following features have not been used: arc inscription, guard expression, CPN/ML statements, fusion places, and functions on the values of the colored tokens. Having such features would result in having smaller, easier to understand, and extendable models.

CPN Tools [3] developed at the University of Aarhus is a graphical ML-based tool for editing and analyzing CP-Nets. Many authors report on projects that investigated the practicability of using CP-nets and the CPN tools for the specification, verification, validation, or performance analysis of the considered system.

Ruilong Wu and Taoshen Li present a checking security protocol method based on CP-Nets [24]. In this method, an intruder model is given and CPN Tools is used. To verify the method, two authentication protocols using asymmetric keys are analyzed. However when the

intruder model is given, the state space explosion problem may follow. For instance, they could not obtain the result at the analysis to TMN protocol, because the state space is too large and there are too many dead markings in their model.

Thus, the attempt to enumerate all meaningful messages that the intruder can send will inevitably lead to an enormous branching of the resulting state space.

D. A Security Protocols Analysis New Method using Colored Petri Nets

Based on the foresaid analysis, we present a new technique using CP-Nets. It is a finite state analysis method. Thus, it involves modeling the protocol as a CP-Net, then an automated tool CPN tools is used to generate all possible states. Insecurities are discovered if an insecure state is reachable in the CPN occurrence graph. Specially, we present a new intruder model.

In protocol analysis, we follow these steps:

1. Build a model of protocol with no intruder: In this step,

(1) Analyzing every sentence of the protocol, and then using CPN ML notation. We declare the color sets, functions, variables, and constants that will be used in the net inscriptions of the CPN model;
(2) Building a communication model in accordance with the order of sending and receiving messages.

2. Add the intruder to the model.

Following the intruder model of Dolev-Yao [2], the intruder has to be modeled with the highest imaginable strength so that all possible attacks on the protocol can be identified. Considering the public channel, the intruder has full control over it. According to the model, he can then carry out the following actions:

(1) Tapping and storage of all messages exchanged via the public channel.
(2) Forwarding, rerouting and blocking of messages.
(3) Generation of forged messages using tapped, randomly generated and obsolete data and encryption techniques.
(4) Decryption of cryptographs if the intruder has a matching key.
(5) The intruder has the ability of a normal principal, so, he can take part in the protocol.

We adopt the assumptions of the Dolev-Yao model. Meanwhile, for overcome the foresaid problem of intruder model, we aim in a less general but complementary approach for the generation of new messages based on an open-ended base of predefined attack tactics. The structure and the number of all possible fake messages are restricted by the patterns and the number of initial messages of the available attack tactics. The intruder model can be thought as two concurrent processes, where the first aims to intercept exchanged messages and the second performs a non-deterministically selected attack tactic against the ongoing protocol sessions (Fig. 2). In this new method, we first give an intruder model to different type of attack. And then, we can attempt to combine multiple attack tactics based on the different attack model, instead of specifying its behavior with a set of rules in Dolev-Yao model.

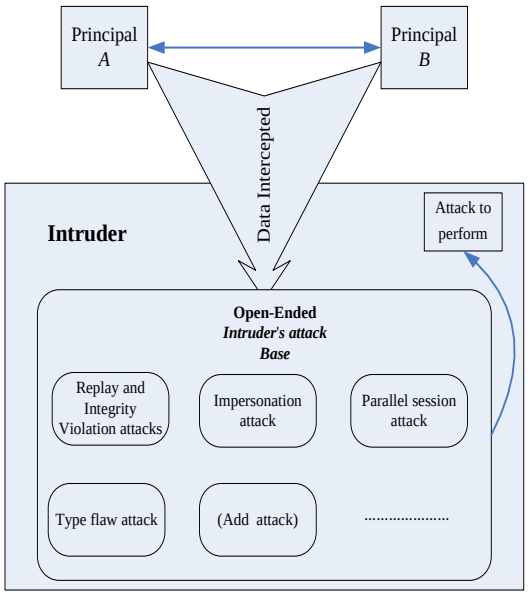

Figure 2. The traditional CP-Nets intruder model.

In this step,

(1) Extending the CPN declarations to include the intruder.
(2) Give an intruder model to every different type of attack tactics. Such as *Replay and integrity violation attacks*, *Type-flaw attacks*, *Impersonation attacks*, and *Parallel session attacks*, and so on. Every intruder model can be integrate into the model of protocol solely which can be modeled checking.
(3) We can attempt to combine multiple attack tactics based on the above attack models.

3. Model checking the CP-Net model of integrated into an intruder.

In the model checking, two methods are used. One exploits the provided state space exploration functions and another is simulation implementation. We run a simulation using the *Simulation tools* of CPN Tools. In the state space exploration check, the steps are as follows:

(1) Enter the state space.
(2) Calculates a strongly connected components (SCC) graph.
(3) Saving a standard state space report.
(4) Make state space queries.
(5) If there are the insecure states, find the possible attacks.

III. ANDREW SECURE RPC PROTOCOL

The best way to explain how a protocol is analyzed using the new method is by example. In this paper, the Andrew secure RPC protocol is specified, and we step through the analysis. In this section, we first introduce our sample protocol, the Andrew secure RPC protocol [6]. It allows two agents, who already share a key K_{AB}, to

agree upon a new session key K'_{AB}, and to perform an authentication handshake.

The Andrew secure RPC protocol is as follows:

1. $A \rightarrow B: A, \{N_a\}_{K_{ab}}$
2. $B \rightarrow A: \{N_a + 1, N_b\}_{K_{ab}}$
3. $A \rightarrow B: \{N_b + 1\}_{K_{ab}}$
4. $B \rightarrow A: \{K'_{ab}, N'_b\}_{K_{ab}}$

Here, principal *A* is a client and principal *B* is a server. N_a and N_b are nonces. N'_b is an initial sequence number which will increase monotonically to be used in subsequent communication. The first message transfers a nonce, which *B* returns in the second message. If *A* is satisfied with the reply, it returns *B*' nonce. After *B* receives and checks the third message, it sends a new session key to *A*.

In [4], this protocol is analyzed using BAN logic, and a weakness is exposed. Further, a correction to the Andrew secure RPC protocol was suggested:

1. $A \rightarrow B: A, N_a$
2. $B \rightarrow A: \{N_a, K'_{ab}\}_{K_{ab}}$
3. $A \rightarrow B: \{N_a\}_{K'_{ab}}$
4. $B \rightarrow A: N'_b$

Although the correction is stronger than the original one, in [5], an attack using two parallel runs of the fixed protocol is found. Gavin Lowe finds an attack on it using two parallel runs. He suggests change the second message to include an encrypted copy of the sender's identity:

$$B \rightarrow A: \{N_a, K'_{ab}, B\}_{K_{ab}}$$

IV. Modeling the Andrew Secure RPC Protocol

In this section, a CP-Net for the Andrew secure RPC protocol fixed in [4] is introduced. Then, an intruder model is developed and integrated into the protocol model. Modeling is performed in CPN Tools.

A. CP-Net Model of the Andrew Secure RPC Protocol

Data is modeled by tokens each belonging to a special data type called the color set of a token. The token color is the actual assignment of values to this token. The CP-Net model for the Andrew secure RPC protocol fixed in [4] is shown in Fig. 3 (declarations) and Fig. 4 (the CPN-Net). In the following, names of places, transitions, and variables of the CPN-Net model are written in italic style.

Declarations:

Referring to Fig. 3, a constant, *u*, is defined to represent the maximum number of the principals in the protocol, another constant, *m*, represents the number of the nonces. Two color sets, *INT*, and *NO*, are declared to the sets of integers. Color set *PART* models the principals. Color set *N* models the nonces. Color set *PROC*, a subset of *PROC1*, models the principals who are performing the protocol. Color set *INT_KK* models the shared key of the two principals. Two color sets, *CRY1*and *CRY2*, are declared for modeling the cryptographs in the message. To model the messages, color sets *MSG1*, *MSG2*, *MSG3*, and *MSG4* are declared. Four color sets, *RUN1*and *RUN 2*, *RUN3*, and *RUN4* are declared for modeling the states of the performing protocol. We declare variables *i*, *j*, *l*, *ln*, and *no*, of type *INT* and *n1* of type *N*.

```
Standard declarations
  val u=2;
  val m=1;
  colset INT = int;
  colset NO=int;
  colset PART = index p with 1..u;
  colset N=index n with 1..m;
  colset PROC1 = product PART*PART;
  fun diff(x,y)=(x<>y);
  colset PROC = subset PROC1 by diff;
  colset K=index k with 1..u;
  colset INT_KK = product INT*K*K;
  colset CRY1=product N*INT_KK*INT_KK;
  colset CRY2=product N*INT_KK;
  colset MSG1=product PART*PART*N;
  colset MSG2=product PART*PART*CRY1;
  colset MSG3=product PART*PART*CRY2;
  colset MSG4=product PART*PART*INT;
  colset RUN1=product PART*PART*N;
  colset RUN2=product PART*PART*N*INT_KK;
  colset RUN3=product PART*PART*N*INT_KK*INT_KK;
  colset RUN4=product PART*PART*N*INT_KK*INT_KK*INT;
  var i,j,l,ln,no:INT;
  var n1:N;
```

Figure 3. Declarations of the CPN-Net model of the Andrew secure RPC protocol.

CP-Net Model:

Fig. 4 shows the CP-Net model of the Andrew secure RPC protocol in the case of normal execution. This model consists of two CP-Net blocks. One CP-Net block describes the *Initiator* of the protocol. The other block describes the *Responder* of the protocol. Principal *p(1)* and *p(2)* may be *Initiator* can also be *Responder*. However, it is impossible that a principal is *Initiator* and also *Responder*, since there is no the token color in color set *PROC*. A time implementation of this CP-Net model related to two principal.

Transitions *Sent1*, *Sent2*, *Sent3* and *Sent4* represent the transmission occurrence of *Message1*, *Message2*, *Message3* and *Message4* respectively. On the other hand, transitions *Rec1*, *Rec2*, *Rec3* and *Rec4* represent the receiving occurrence of *Message1*, *Message2*, *Message3* and *Message4* respectively. Places *Run1_1*, *Run1_2*, *Run1_3*, and *Run1_4* hold the state after the implementation of every step of *Initiator*. Place *Run2_11*, *Run2_12*, *Run2_2*, *Run2_3*, and *Run2_4* hold the state after the implementation of every step of *Responder*. After the protocol carries out, *Run1_4* and *Run2_4* save the tokens which can prove that the protocol has already carried out. Places *M1*, *M2*, *M3*, and *M4* hold the state of the message in the public channel. Thus, the intruder can modify the messages, replay the messages or pass the messages without any modifications.

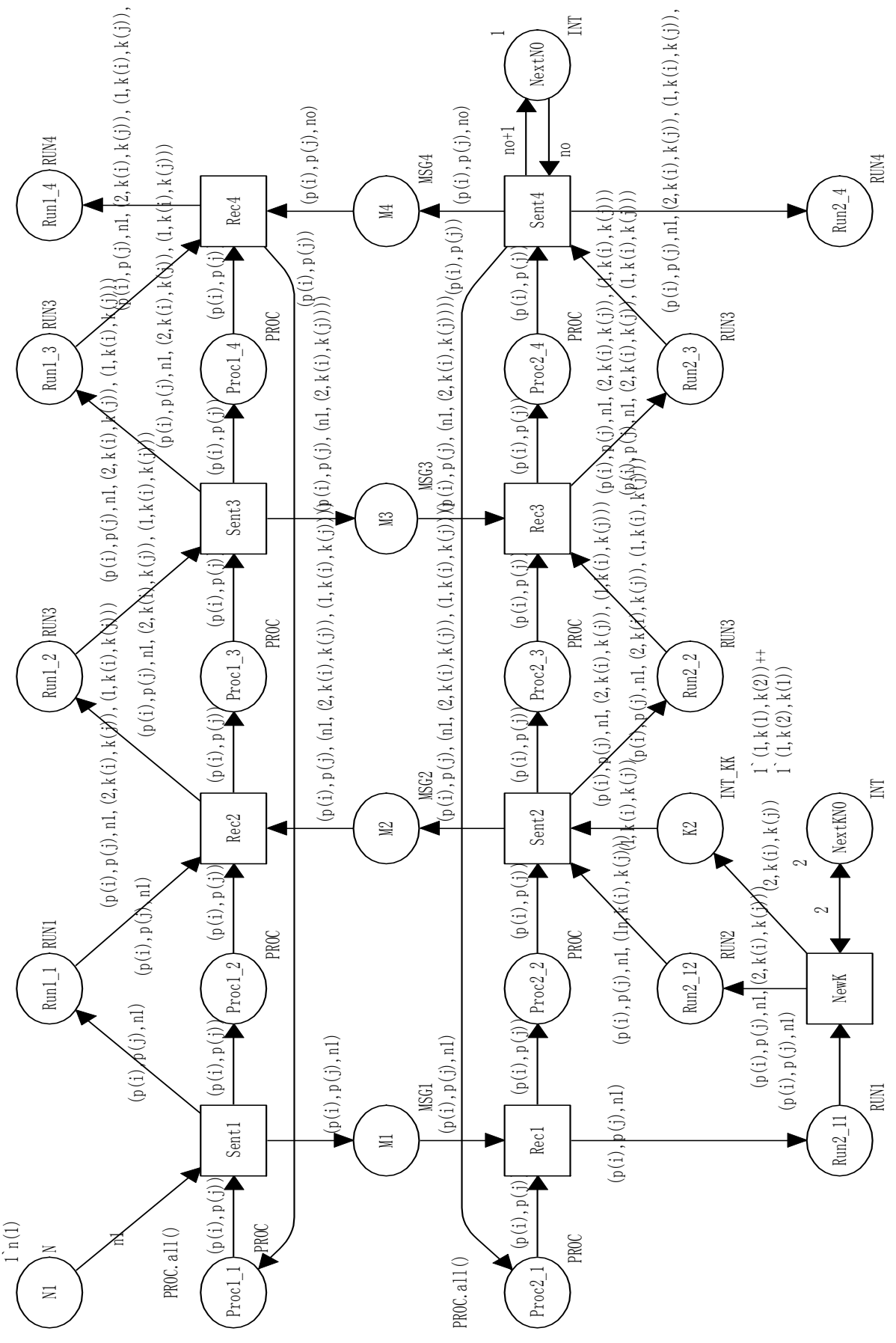

Figure 4. CP-Net model of the Andrew secure RPC protocol

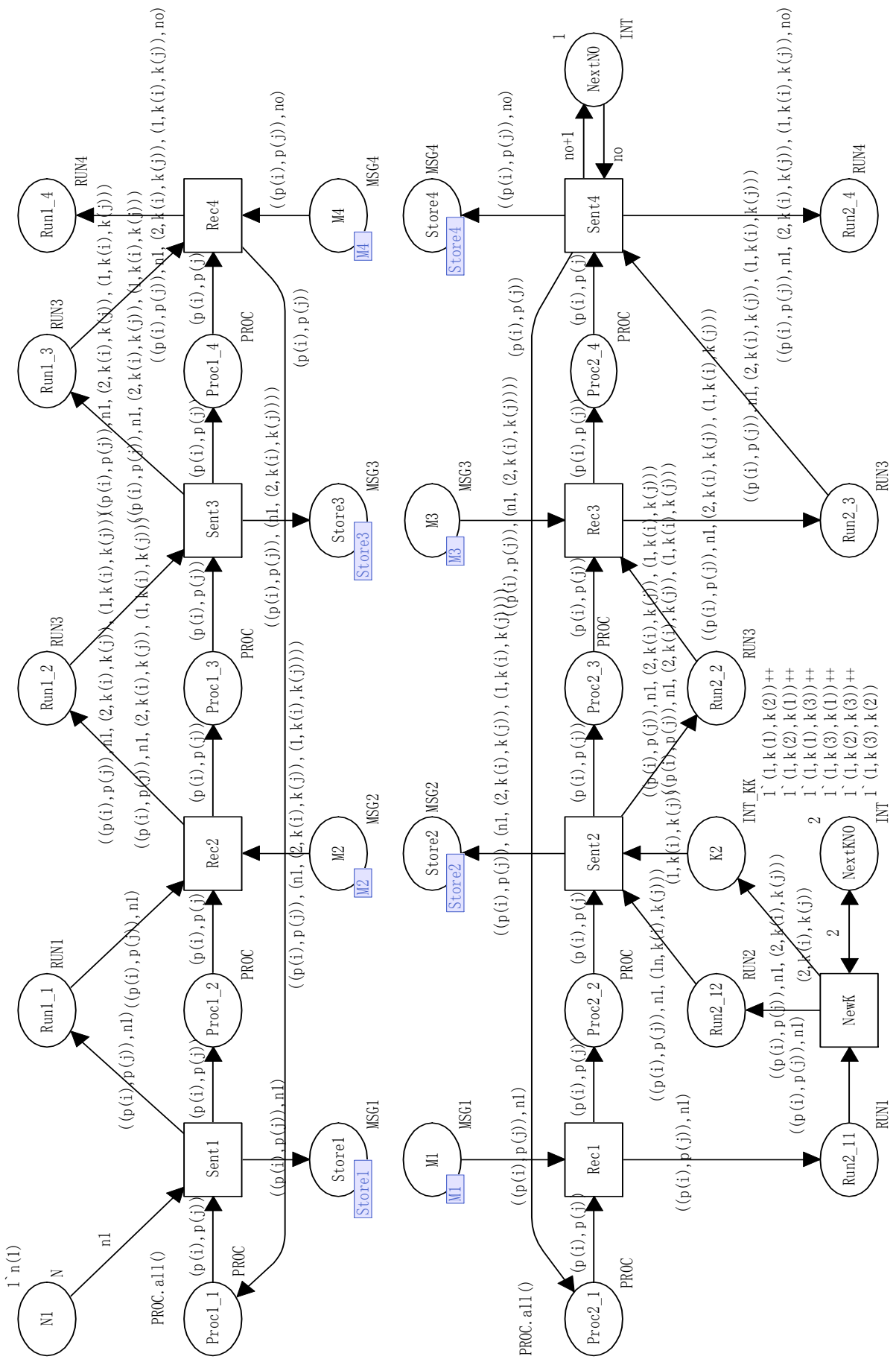

Figure 5. CP-Net model of the Andrew secure RPC protocol (with an intruder)

V. ANALYZING THE ANDREW SECURE RPC PROTOCOL

In this section, model checking of the two CP-Net models is performed in CPN Tools. In the model checking, two methods are used. One exploits the provided state space exploration functions and another is simulation implementation. After model checking and the state space analysis, an attack is found which the same as the one found by Gavin Lowe [5].

A. State Space Analysis of the Andrew Secure RPC Protocol CP-Net Model

To analyze the desired properties of the Andrew secure RPC protocol, we firstly check the state space standard report generated by CPN Tools.

A part of the report is listed below:

```
Statistics
------------------------------------------------------------------------
  State Space
    Nodes:  19
    Arcs:   18
    Secs:   0
    Status: Full
  Scc Graph
    Nodes:  19
    Arcs:   18
    Secs:   0
Home Properties
------------------------------------------------------------------------
  Home Markings
    None
 Liveness Properties
------------------------------------------------------------------------
  Dead Markings
    [18, 19]
  Dead Transition Instances
    None
  Live Transition Instances
    None
```

The report shows that a full state space with 19 nodes and 18 arcs is generated. We also found 2 dead markings in the state space, which are nodes18, 19. We can use the state space exploration functions shown in Fig. 7 to know the tokens in the places *Run1_4* and *Run2_4* of node 18. From Fig. 7 we can know that the *Initiator* is *p(1)* and the *Responder* is *p(2)* in this implementation of the protocol. In addition, *p(1) and p(2)* have the same believes. There is a similar result to node 19.

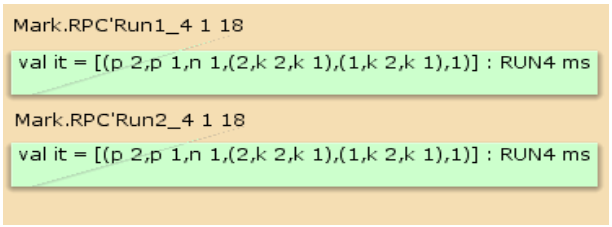

Figure 7. Function *Mark* and the result 1.

B. State Space Analysis of CP-Net Model of the Andrew Secure RPC Protocol Integrated into an Intruder Model

In the same way, we firstly check the state space standard report of CP-Net model integrated into an intruder model.

A part of the report is listed below:

```
Statistics
------------------------------------------------------------------------
  State Space
    Nodes:  79
    Arcs:   78
    Secs:   0
    Status: Full
  Scc Graph
    Nodes:  79
    Arcs:   78
    Secs:   0
Home Properties
------------------------------------------------------------------------
  Home Markings
    None
 Liveness Properties
------------------------------------------------------------------------
  Dead Markings
    6 [79, 78, 77, 76, 75, ... ]
  Dead Transition Instances
    None
  Live Transition Instances
    None
```

The report shows that a full state space with 79 nodes and 78 arcs is generated. It is pleasing to see that the state spaces are comparatively smaller. We only found 6 dead markings in the state space. However, the report does not give the all dead markings. Thus, the function ListDeadMarkings( ) is used, and the result is shown in Fig. 8.

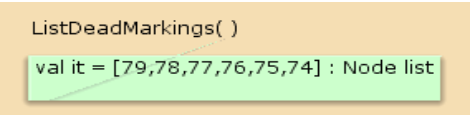

Figure 8. Dead markings for the described CP-net.

Additionally, we use the state space exploration functions shown in Fig. 9 to know the tokens in the places *Run1_4* and *Run2_4* of node 79. From Fig. 9, we can see that *p(1)* and *p(2)* do not have the same believes.

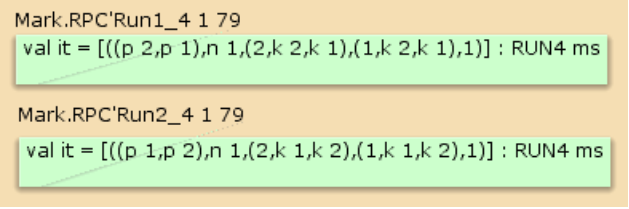

Figure 9. Function *Mark* and the result 2.

To get more details, we run a simulation to the CP-Net. Variable *i* is bound to 1 and *j* is bound to 2. After the simulation, we get the following implementation sequence. Each line in the occurrence sequence represents a step that has a single binding element. Each line contains the following information: the page name, the transition, and the binding. For instance, the line identified by (*), on its right side, represents the step (*Sent4* in the *RPC*, no = 1, n1 = n (1), j = 1, i = 2).

RPC Sent1 n1 = n(1), j = 2, i = 1
Intruder Replay1 n1 = n(1), j = 2, i = 1
RPC Rec1 n1 = n(1), j = 1, i = 2
RPC NewK n1 = n(1), j = 1, i = 2
RPC Sent2 n1 = n(1), j = 1, ln = 2, i = 2
Intruder Replay2 n1 = n(1), j = 1, i = 2
RPC Rec2 n1 = n (1), j = 2, i = 1
RPC Sent3 n1 = n(1), j = 2, i = 1
Intruder Replay3 n1 = n(1), j = 2, i = 1
RPC Rec3 n1 = n(1), j = 1, i = 2
RPC Sent4 no = 1, n1 = n(1), j = 1, i = 2 (*)
Intruder Replay4 no = 1, j = 1, i = 2
RPC Rec4 no = 1, n1 = n(1), j = 2, i = 1

This attack is stated in a high level description as follows, noting that I_B denotes *I* impersonating *B*. Thus, a step of the form "$I_B \to A: X$" means that *I* poses as *B* and sends *X* to *A*, whereas a step of the form "$A \to I_B : X$" means that *I* intercepts the message *X* originally sent from *A* to *B*. This attack involves two separate runs of the protocol labeled *I* and *II*.

*I*1. $A \to I_B : A, N_a$

II 1. $I_B \to A: B, N_a$

II 2. $A \to I_B : \{N_a, K_{ab}^{'}\}_{K_{ab}}$

*I*2. $I_B \to A: \{N_a, K_{ab}^{'}\}_{K_{ab}}$

*I*3. $A \to I_B : \{N_a\}_{K_{ab}^{'}}$

II 3. $I_B \to A: \{N_a\}_{K_{ab}^{'}}$

*I*4. $I_B \to A: N_i$

II 4. $A \to I_B : N_a^{'}$

In this sequence, *A* (In the CP-Net is *p(1).*) believes that he has established a session with *B* (In the CP-Net is *p(2).*), and he believes that *B* has established a session with him, even though *B* may in fact be absent. There are the similar results to the other nodes.

Coincidentally, this attack is the same as the one found by Gavin Lowe [5]. In fact, we have analyzed the original protocol and its fixes in [4] and [5] using Rubin logic, and we also present a new fix in which the weakness no longer exists [7].

VI. Conclusion

Based on the current security protocol analysis techniques using CP-Nets, we have presented a new and promising method that uses CP-Nets for the analysis of security protocols. The main contribution of this work is a CP-Net intruder model that provides an open-ended base for the integration of multiple attack tactics. This is a viable approach to overcome the state space explosion problem. But also, the presented intruder model is open to integrate more specialized attack. Furthermore, we show how to use CPN Tools in the construction and model checking of the net models. Finally, we use the new method to model and analysis the Andrew secure RPC protocol fixed in [4].

In the example, we a CP-Net for the Andrew secure RPC protocol fixes in [4] has been presented. And then an intruder model is developed and integrated into the protocol model. Model checking is performed in CPN Tools. In the model checking, two methods are used. After model checking and the state space analysis, an attack is found which the same as the one found by Gavin Lowe. For the sake of simplicity, we use the hierarchal CP-Nets in our analysis of the protocol.

In the future, we would like to formalize the different type of attack tactics and give more exact CP-Nets models of intruder, and to use the new method to analyze other security protocols. We are also interested in the other state space reduction methods to overcome the state space explosion problem in the specialized CP-Nets.

Acknowledgment

The Authors thank Professor Kaile Su of Peking University for his excellent suggestion to pursue this topic. The authors wish to thank Professor Qihai Zhou and the anonymous reviewers for their comments that helped increase the quality of the paper. This work was supported in part by National High Technology Research and Development Program (863 Project) of China under Grant 2007AA010609, in part by Key Program of National Natural Science Foundation of China under Grant 90718009, in part by Science and Technology Foundation of Guizhou Province, China under Grant 20082125.

References

[1] K. Jensen, *Coloured Petri Nets: basic concepts, analysis methods and practical use*, Vol. 1–3, Monographs in Theoretical Computer Science, Springer-Verlag, 1997.
[2] D. Dolev, and A.Yao, "On the security of public key protocols," in *Proc. 1981 IEEE Symposium on Foundations of Computer Science*, pp. 350–357.
[3] CPN Tools Homepage: http://wiki.daimi.au.dk/cpntools/cpntools.wiki.
[4] M. Burrows, M. Abadi, and R. Needham, "A logic of authentication," *ACM Transactions on Computer Systems*, vol. 8, no. 1, 1990, pp. 18–36.
[5] Gavin Lowe, "Some new attacks upon security protocols," *in Proc. 9th 1996 IEEE Workshop Computer Security Foundations*, pp. 162–169.
[6] M. Satyanarayanan, "Integrating security in a large distributed system," *ACM Transactions on Computer Systems*, vol. 7, no. 3, 1989, pp. 247–280.
[7] Yang Xu, and Xiaoyao Xie, "Analysis of authentication protocols based on Rubin logic," *in Proc. 4th IEEE Int.*

Conf. Wireless Communications, Networking, Mobile Computing, 2008, pp. 1–5.
[8] Girault Claude, and Rudiger Valk, *Petri nets for systems engineering: a guide to modeling, verification, and applications*, Springer-Verlag, 2003.
[9] C. S. Calude, Gheorghe Paun, Grzegorz Rozenberg, and Arto Salomaa, *Multiset processing: Mathematical, computer science, and molecular computing points of view*, LNCS 2235, Springer-Verlag, 2002, pp. 347–358.
[10] C. Capellmann, S. Christensen, and U. Herzog, "Visualising the behaviour of intelligent networks," *Services and Visualization, Towards User-Friendly Design*, LNCS1385. Springer-Verlag, 1998, pp. 174–189
[11] C. Capellmann, H. Dibold, and U. Herzog, "Using high-level Petri nets in the field of intelligent networks," *Application of Petri Nets to Communication Networks*, LN CS1605, Springer-Verlag, 1999, pp. 1–36.
[12] S. Christensen and J. Jørgensen, "Analysing Bang & Olufsen's BeoLink Audio/Video system using coloured Petri nets," In *Proc. 18th International Petri Net Conference*, 1997, pp.387–406.
[13] J. de Figueiredo and L. Kristensen, "Using coloured Petri nets to investigate behavioural and performance issues of TCP protocols," In *Proc. 2nd Workshop on Practical Use of Coloured Petri Nets and Design/CPN*, Aarhus, *1999*, pp. 21–40.
[14] L. Kristensen, J. Jørgensen, and K. Jensen, "Application of coloured Petri nets in system development," In *Proc. 4th Advanced Course on Petri Nets, LNCS3098*, Springer-Verlag, 2004, pp. 626–685.
[15] Yang Xu, and Xiaoyao Xie, "Modeling and Analysis of an Online Score System Using Colored Petri Nets," In *Proc. 3rd International Conference on Anti-counterfeiting, Security, and Identification*, Hong Kong, 2009, pp: 432–436.
[16] E. Doyle, S. Tavares, and H. Meijer, "Automated security analysis of cryptographic protocols using Coloured Petri Net specifications," *Workshop on Selected Areas in Cryptography, SAC'95 Workshop Record*, 1995, pp. 35–48.
[17] E. Doyle, S. Tavares, and H. Meijer, "Computer analysis of cryptographic protocols using Coloured Petri Nets," *18th Biennial Symposium on Communication*, Kingston, Ontario, 1996, pp. 194–199.
[18] A. M. Basyouni, *Analysis of wireless cryptographic protocols*, Master's Thesis, Queen's University Kingston, Ontario, Canada, 1997.
[19] 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*, 1994, pp. 107–118.
[20] HeeChul Moon, *A study on formal specification and analysis of cryptographic protocols using Colored Petri Nets*, Master's Thesis, Kwangju institute of science and technology, Korea, 1998.
[21] Daobin Liu, Li Guo, and Shuo Bai, "Formal analysis of security protocol using Petri Nets," *Acta Electronica Sinica*, vol.32, no.11, 2004, pp. 1926–1929.
[22] W. Dresp, *Computer-gest ützte Analyse von kryptographischen Protokollen mittels gef ärbter Petrinetze*, Diploma Thesis, Department of Business Information Systems, University of Regensburg, 2004.
[23] S. Aly and K. Mustafa, *Protocol verification and analysis using colored Petri nets*, Technical Report TR-04-003, DePaul University, 2004.
[24] Ruilong Wu, *Research on checking security protocols technology based on CPN models*, Master's Thesis, Guangxi University, Nanning, China, 2005.

Yang Xu is a Ph. D. candidate, major in the software and theory of computer, school of computer science and technology, Guizhou University, China. He is a member of Chinese Association for Cryptologic Research. He is the author of more than 10 journal and international conference papers. During 2007-now, he stays in Key Laboratory of Information and Computing Science of Guizhou Province, China to study network and information security, and protocol analysis.

Xiaoyao Xie was born in Guiyang, China in 1952. He received his Ph.D degree from Wuhan University, Wuhan, China. He is a professor and Ph. D supervisor. He is a member of Chinese Association for Cryptologic Research and a member of IEEE. His research interests include network and information security, and protocol analysis.