Protocol Verification And Analysis Using Colored Petri Nets

Technical Report
Submitted By

Salah Aly
DePaul University
aly@cs.depaul.edu

Khaled Mustafa
Cairo University
kelsayed@ntgclarity.com

July, 2003

Contents

Chapter 1

Protocol Verification And Analysis

In the world of designing security protocols, verification is a crucial step to eliminate weaknesses and inaccuracies of effective security protocols. There are many models and tools to verify security protocols, including, Finite State Machines (FMS), Colored Petri Nets (CP-Nets), Cryptographic Protocol Analysis Language Evaluation System (CPAL-ES), BAN Logic, Morphi, etc. In this work, we study CP-Nets model and show how it can be used to analyze security protocols. As an example, we model and verify Station-to-Station protocol (STS) using CP-Nets.

1.1 Verification Tools and Models

The most known tools for protocol verifications are Petri Nets [10], Finite State Machines [6] , CPAL-ES [15].

- **Petri Nets:** The history of Petri Nets goes back to the work of Carl Adam Petri during his Ph.D. thesis in Germany in 1962. A Petri Net is a graphical and mathematical tool to verify systems and protocols. Many of researchers have used Petri Nets to analyze and verify systems in different areas of science such as artificial intelligence, parallel processing system, control systems, and numerical analysis.
  Petri Nets in the graphical forms are like flowcharts and network diagrams, while in mathematical forms, they are like algebra and logic subjects.
  Although Petri Nets have existed for many decades, they have been recently used to verify cryptographic and security protocols. And the most known research centers that have used these Nets are Cambridge [10], Aarhus [6], and Queen's Universities [8].

- **Finite State Machines (FSM):** Finite state machines mainly consist of a set of transition rules. In the traditional finite state machines model, the environment of the machine consists of two finite and disjoint sets of signals, input signals and output signals. Also, each signal has an arbitrary range of finite possible values [6].

- **CPAL-ES:** CPAL-ES is a formal cryptographic protocol evaluation system. In CPAL-ES, Protocols are expressed in CPAL Language, which includes the assumptions, principles, actions, and goal of the protocols run. The system returns a verification condition (VC). CPAL-ES is a three-step process of encoding of the protocol into CPAL, translating the specification into a VC, and proving the VC. A lot of research work was done in this area, to be mentioned here Ph. D. Thesis of Professor Alec F. Yasinsac and Ph.D. Thesis of Professor Brett Tjaden at University of Virginia [15].

1.2 Protocol Analysis Using Colored Petri Nets

There are two forms of Petri Nets: ordinary Petri Nets and high level Petri Nets.
In the ordinary Petri Nets, a system can be modelled by a graph, which has two kinds of nodes, places and transitions. Each place (circle) is connected with a transition (rectangle) by arcs. The distribution of tokens, which are groups of black dots located in places, is called a marking, which represents the current state of the net [9], [2].

1.2.1 Usage of Colored Petri Nets

In the high level Petri Nets; such as CP-Nets, Predicated and Transition Nets, and Numerical Petri Nets, each token can hold and represent different information and data.
In addition, there are two types of CP-Nets: non-hierarchical CP-Nets and hierarchical CP-Nets [14]. However, what we are using in this study is the Hierarchical CP-Nets.

- Graph Colored Petri Nets
  Colored Petri Nets, which can be used in a graph, have four essential elements; places, transitions, arcs, and tokens. As finite state machines and other tools, CP-Nets are used to detect protocol failures as we explain in this Chapter.

- Algebraic Colored Petri Nets
  It is another form of CP-Nets that represents a system or a protocol as a grammar language in an algebraic form.

1.2.2 Incidence Matrix (State Equation)

As stated in [7], the dynamic behavior of many systems and protocols can be described by differential or algebraic equations.
The incidence matrix A for a CP-Nets N with m transitions and n places is defined as $A = [a_{ij}]$ an $n \times m$ matrix of integers and its typical entry is given by

$$a_{ij} = a_{ij}^{+} - a_{ij}^{-}$$

Where $a_{ij}^{+} = w(i,j)$ is the weight of the arc from transition i to its output place j and $a_{ij}^{-} = w(i,j)$ is the weight of the arc to transition i from its input place j.
a_{ij}^{+}, a_{ij}^{-} , and a_{ij} represent the number of tokens removed, added, and changed in place j when transition i fires once.

Since the ith row of the incidence matrix A denotes the change of the marking as the result of firing transition i, the state equation can be written as follows:

$$M_k = M_{k-1} + AX$$

Where k=1,2,3,., and X is the firing vector as described in [7],[10].

1.3 Why Use Colored Petri Nets?

There are many useful usages of Colored Petri Nets. Kurt Jensen wrote a powerful paper on theoretical aspects of CP-Nets, he stated 13 reasons for using CP-Nets. We mention them briefly and for further detail, refer to Jensen's papers [7], [8]:

- *CP-Nets have a graphical representation*
- *CP-Nets have a well-defined semantics, which unambiguously defines the behavior of each CP-Nets*
- *CP-Nets are very general and can be used to describe a large variety of different systems*
- *CP-Nets have very few, but powerful, primitives*
- *CP-Nets have an explicit description of both states and actions*
- *CP-Nets have a semantics that builds upon true concurrency, instead of interleaving*
- *CP-Nets offer hierarchical descriptions*
- *CP-Nets integrate the description of control and synchronization with the description of data manipulation*
- *CP-Nets can be extended with a time concept*
- *CP-Nets are stable towards minor changes of the modelled system*
- *CP-Nets offer interactive simulations where the results are presented directly on the CPN diagram*
- *CP-Nets have a large number of formal analysis methods by which properties of CP-Nets can be proved*
- *CP-Nets have computer tools supporting their drawing, simulation and formal analysis*

1.4 Colored Petri Nets Objects

Figure 1.1 shows that each entity can be represented as a side in the Petri Nets. In the case of client-server, there are two main sides representing them. The flow of input and output is portrayed in a form of arcs and lines between the two sides. These arcs contain the colored data [3]. Figure 1.2 presents the intruder model between the client and the server. The intruder can modify the transaction or pass the messages without any modifications.

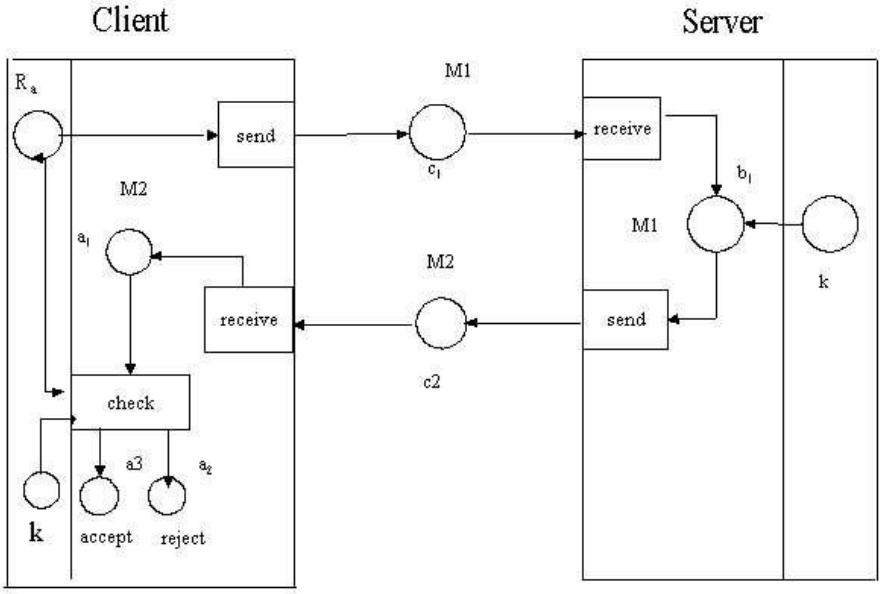

Figure 1.1: Introductory CP-Nets model

1.5 Modelling Protocols Using Colored Petri Nets

In the last decade, lots of research to represent security and cryptographic protocols were done. Thus, the group of cryptography at Queen's University and Computer Laboratory at University of Cambridge added significant research contributions to verify cryptographic and security protocols, compute their weaknesses using CP-Nets as well.

There are two courses for using CP-Nets: forward or backward analysis. As Ayda and Moon stated [3], [9], the backward state analysis of a cryptographic protocol includes three steps:

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

Our model mainly depends on their work with some minor changes and improvements: The model consists of the following steps:

- Step1: describe the protocol in a CP-Nets form
- Step2: write Acceptance Check Steps (ACS)
- Step3: describe the intruder model

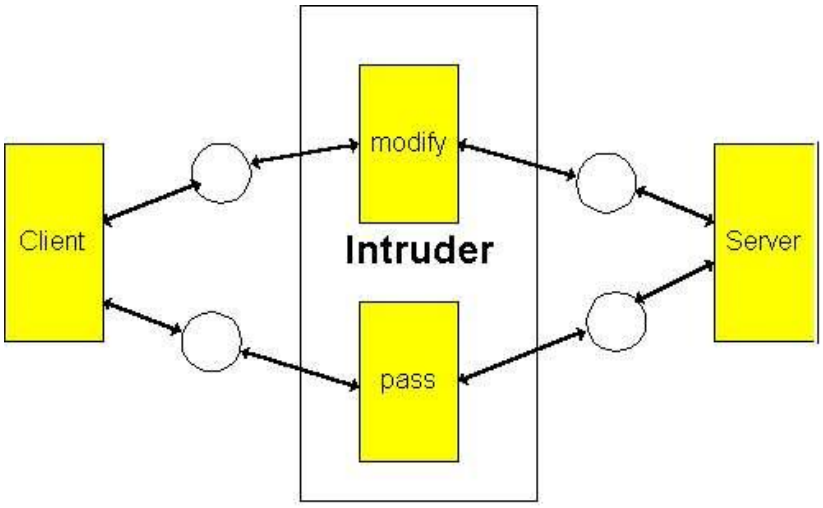

Figure 1.2: CP-Nets intruder model

- Step4: find the insecure states
- Step5: apply the Matrix Analysis Steps (MAS), and then run the implemented computer program to solve the equation

$$M_n = M_0 + A\sum_{i=1}^{m}\sigma_i^t$$

Where M_n is the insecure state, M_0 is the initial marking, A is the matrix description, and σ^t is a transpose vector, which determines the firing states.

Remarks:

- The Matrix Analysis Steps (MAS)

$$M_n = M_0 + A\sum_{i=1}^{m}\sigma_i^t$$

If we know the initial marking and the insecure state, Then by the matrix description, we can solve the above equation to test if the insecure states exist or not.

- The values of the vector σ^t are only 0's and 1's corresponding to not fired, fired of each transition. In the general case, every transition may fire more than once. If each transition is allowed to fire only once (as our STS model described in next chapter) the above equation may be simplified, as Ayda mentioned [4], to:

$$M_n = M_0 + A\sigma^t$$

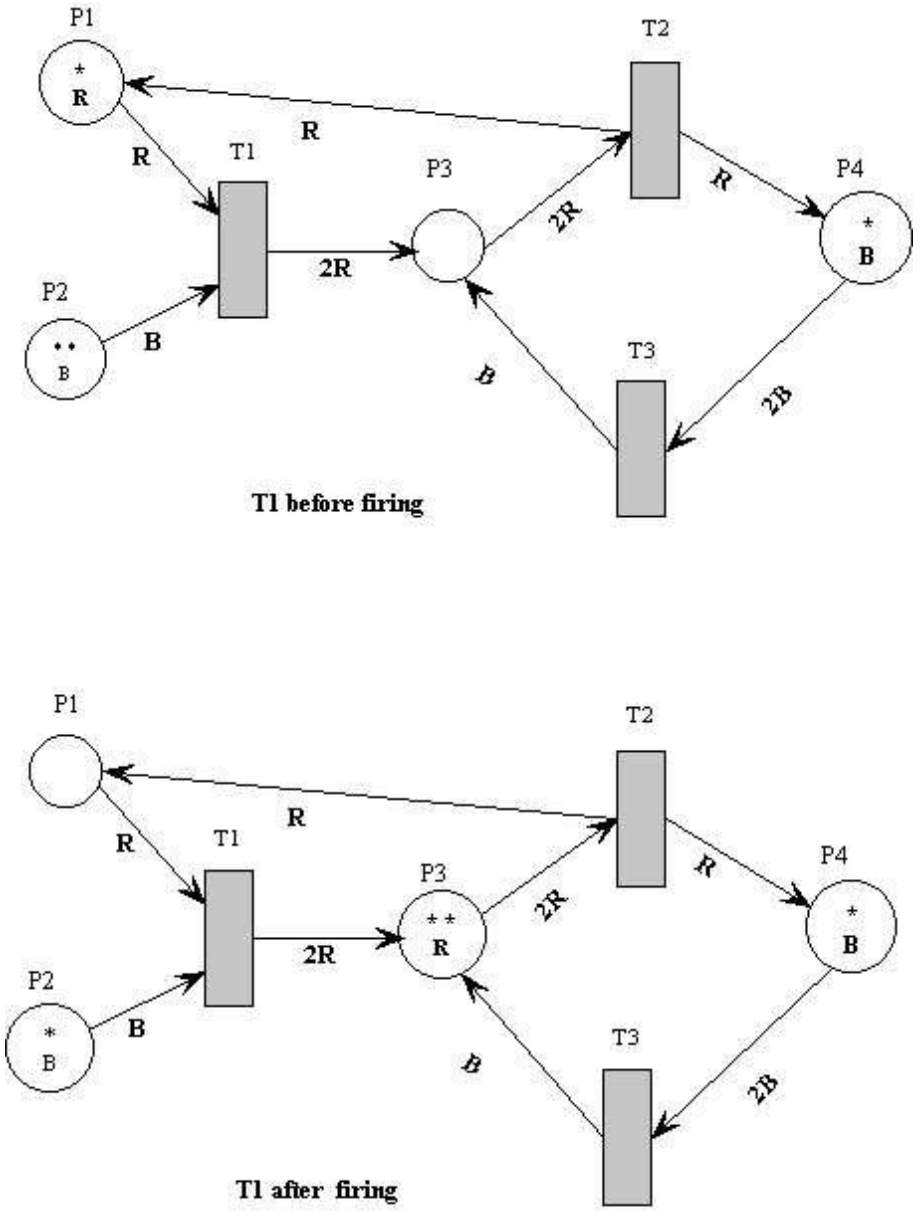

Figure 1.3: Example of CP-Nets

Example:

Suppose the initial marking $M_0 = \begin{pmatrix} R \\ 2B \\ 0 \\ B \end{pmatrix}$

and the matrix description of the Figure 5.3 is $A = \begin{pmatrix} -R & R & 0 \\ -B & 0 & 0 \\ 2R & -2R & B \\ 0 & R & -2B \end{pmatrix}$

and $\sigma = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$

Then we can compute M_n as follows:

$$M_1 = \begin{pmatrix} R \\ 2B \\ 0 \\ B \end{pmatrix} + \begin{pmatrix} -R & R & 0 \\ -B & 0 & 0 \\ 2R & -2R & B \\ 0 & R & -2B \end{pmatrix} * \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ B \\ 2R \\ B \end{pmatrix}$$

1.6 MAS Flow Chart and Implementation

This flow chart, in Figure 5.4, supposes we know the insecure state and want to verify whether the vector σ^t exists or not. Moreover, this is what actually is happening in verification of security protocols using CP-Nets as in our model and approach. Where, m is the number of transitions and n is the number of places.

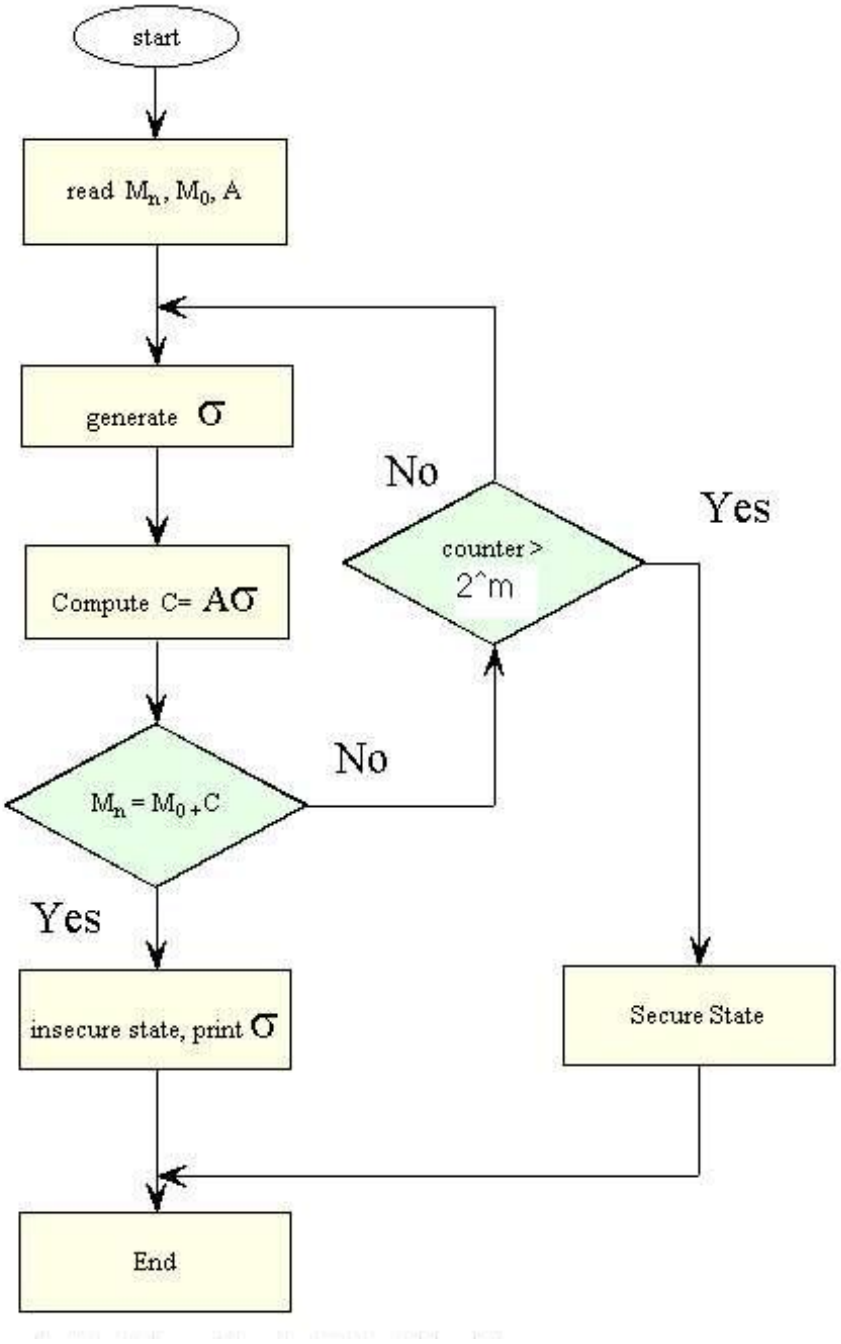

Figure 1.4: Flow chart of MAS model

Chapter 2

Verification And Analysis of STS Protocol Using Colored Petri Nets

In this chapter we demonstrate our model as introduced in the previous chapter to analyze and verify Station-To-Station (STS) protocol using Colored Petri Nets (CP-Nets).

2.1 STS Protocol Overview

STS is a key agreement protocol. In contrast to all key agreement protocols, STS provides both mutual entity authentication and mutual explicit key authentication. According to the previous survey of web security protocols, they do not support or utilize STS. We try to analyze STS in order to evaluate its security level. Therefore, as a future work, web security protocols may support STS protocol.

Protocol Scenario:

- STS initially is based on Diffie-Hellman protocol in the first message from entity A to entity B or from client to server. The entity A generates a random value x and compute the term $\alpha^x mod P$ where both α and P are known integers values for the two entities.
- In second message of negotiation, entity B generates a random value y, computes the shared key K. It sends an encrypted message enciphered with the computed key. The encrypted message itself is a signed message of B's private key. Upon entity A receiving the second message, the shared key is computed and it can verify that the signed message is from entity B after decrypting it with B's public key embedded in the message.
- Finally, the entity A signs a message with its private key. Then it sends the message encrypted with the shared key K as in Fig 2.1.

Protocol Messages:

1- $A \longrightarrow B : A, \alpha^x mod P$

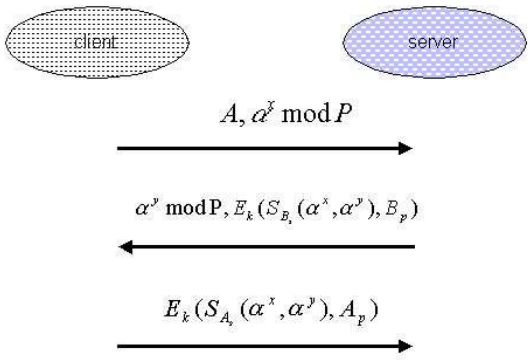

Figure 2.1: STS exchange messages

2- $A \longleftarrow B : \alpha^y mod P, E_k(S_{B_s}(\alpha^x, \alpha^y), B_p))$

3- $A \longrightarrow B : E_k(S_{A_s}(\alpha_x, \alpha_y), A_p)$

Note that:

- The signature in the second and third messages is based on public key cryptography. In some cases, instead of using the signature with PKCS [1], we use MAC [9],[4] or one-way hash function of the key k within the signed message.
- It is true that the computed key between A and B is strong enough if the exponential factors x and y are cryptographically strong long random numbers.

2.2 Modelling and Analyzing STS using CP-Nets

In this section, the STS is described using Colored Petri Nets (CP-Nets) [2].

Steps of Analysis:

Step 1: model the STS using CP-Nets illustrated in the Figure 2.2.

$M_1 : A, \alpha^x mod P$

$M_2 : \alpha^y mod P, E_k(S_{B_s}(\alpha^x, \alpha^y), B_p)$

$M_3 : E_k(S_{A_s}(\alpha_x, \alpha_y), A_p)$

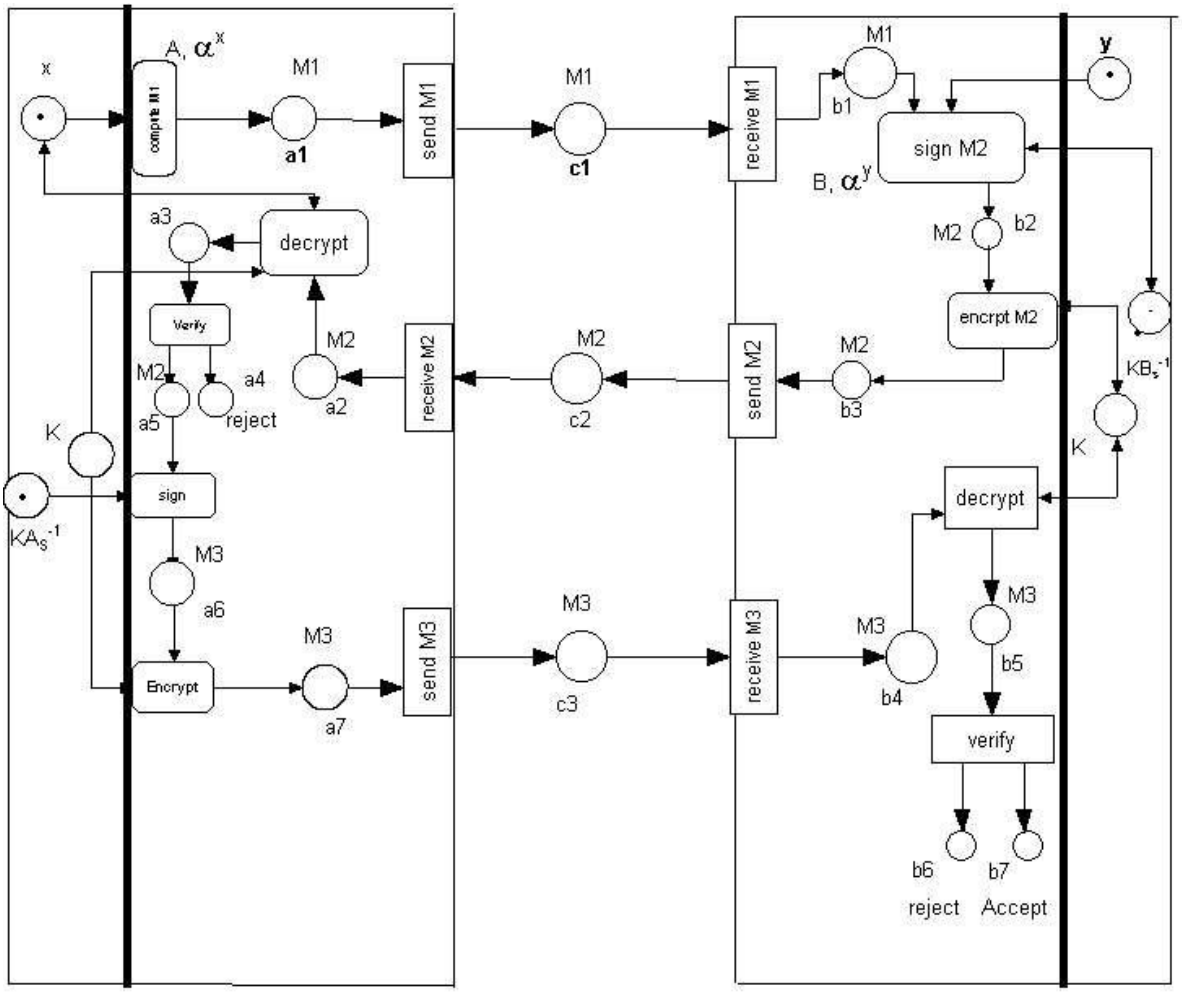

Figure 2.2: CP-Nets model for STS protocol

Step 2: apply the Acceptance Check Step (ACS) to STS messages. We note that the general man-in-middle intruder can exist between the two entities or client-server. As the intruder works in the model, we forward to the step3. It is clear that the intruder cannot get the shared key between the two entities, but s/he owns a key that can be shared in the secure transition between the two sides. The weakness of this model depends mainly on that both sides do not verify each other or there is no certification for the entities.

Step 3: add the proposed intruder side in the model as in the Figure 2.3.

$M_1 : A, \alpha^x modP$

$M_1^{\backslash} : A, \alpha^z modP$

$M_2 : \alpha^y modP, E_{k2}(S_{B_s}(\alpha^z, \alpha^y), B_p)$

$M_2^{\backslash} : \alpha^z modP, E_{k1}(S_{B_s}(\alpha^z, \alpha^x), B_p)$

$M_3 : E_{k1}(S_{A_s}(\alpha_x, \alpha_z), A_p)$

$M_3^{\backslash} : E_{k2}(S_{A_s}(\alpha_z, \alpha_y), A_p)$

At this point, there are two parts of analysis for the intruder model. Part I is a specification of STS protocol and Part II is the case where the intruder can attack the protocol.

Specifying STS Protocol: Part I

Step 4.1: By analyzing the protocol as in Figure 2.3, we find that man-in-middle attack has the ability to direct the negotiation between the client and server. The intruder shares K_1 with the client and K_2 with the server.

Step 5.1: As Figure 2.4 illustrates, the dashed rectangle is the first part of the matrix description, so the initial making can be defined as

$M_0^t = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$, where n = 20 and the first element in the vector corresponds to a_1, the second element corresponds to a_2, etc.

And the insecure state

$M_n^t = (0, 0, 0, 0, M_2^{\backslash}, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$, where n = 20.

And the Matrix description A exists in table 1.1.

Applying MAS, we find that the defined final insecure state is reachable from the initial state, which is considered a major problem in the security of the STS protocol. Also, we found that every transition fires at most once to get to the insecure state.

To avoid this problem, Wilson and Menezes suggested the following in their paper [5]:

first, adding certification for the outgoing message to get the public key for each entity: this would be better than getting it from the decryption of the second and the third messages using the shared key. Second, adding the identity of each entity within the signature.

$M_1 : A, \alpha^x mod P$

$M_2 : Cert(B), \alpha^y mod P, E_k(S_{B_s}(\alpha^x, \alpha^y, A, B), B_p)$

$M_3 : Cert(A), E_k(S_{A_s}(\alpha_x, \alpha_y A, B), A_p)$

Note: as the part I of the CP-Nets for the intruder have been analyzed. The part II (the rest of the model) can also be analyzed and the same results could be reached.

Specifying STS Protocol: Part II

Step 4.2: by analyzing the protocol we find that man-in-middle attack has the ability to control the negotiation between the client and the server. The intruder shares K_1 with the client and K_2 with the server.

Step 5.2: As Figure2.5 illustrates the dashed rectangle is the first part of the matrix description and the initial marking is

$M_0^t = (M_2^{\backslash}, 0, K_{AI}, 0, 0, 0, KBI, 0, 0, 0, 0, K_{AI}, 0, 0, K_{BI}, 0, 0)$, where n = 17.

And the insecure state

$M_n^t = (0, 0, 0, 0, 0, 0, 0, 0, 0, M_3^{\backslash}, M_3^{\backslash}, 0, 0, 0, 0, 0, 0)$, where n =17.

For the description of matrix A, refer to Table 1.2 (Appendix A).

Applying MAS, we find that the defined final insecure state is reachable from the initial state. Also, we found that every transition fires at most once to get the insecure state.

2.3 Intruder Model Between the Client and the Server

Figure 2.3 illustrates an intruder between a client and a server. We observe that the intruder can modify the outgoing messages from the client to the server and vice versa. We study the case of man-in–middle attack, although different attack models can be applied to the STS protocol.

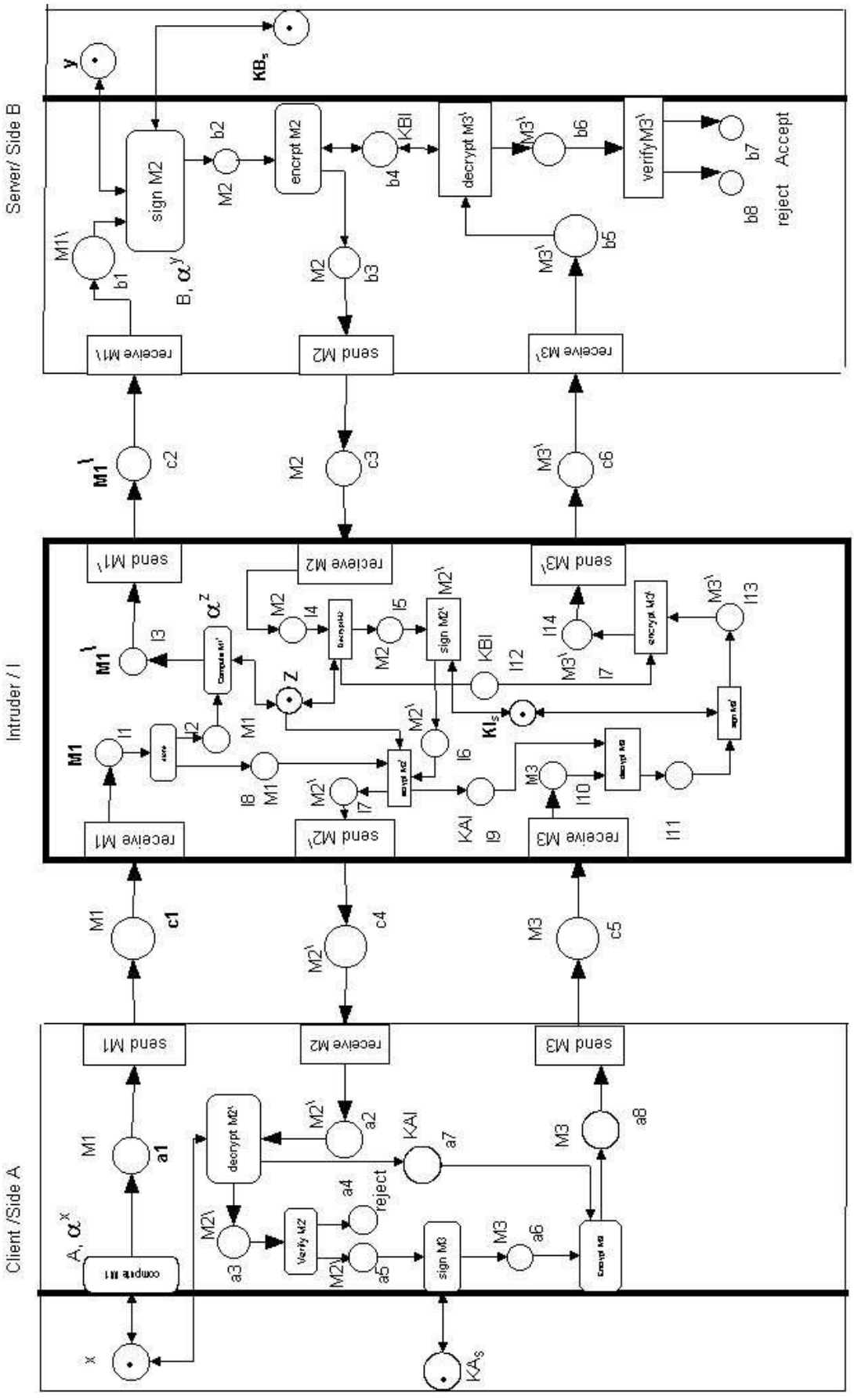

Figure 2.3: CP-Nets model for STS-intruder "basic case"

2.4 Analysis of STS with CP-Nets

From Figures 2.5 and 2.4, the intruder can intercept the transmission between the client and server. Protocol analyses and verifications assume that we can detect and know the intruder behavior between the client and server. Moreover, they suppose that the weaknesses and correctness of a protocol can be shown. In the case of STS protocol, there are two cases for checking and verifying as we mention.

The protocol and the attack are explained below:

- The intruder I intercepts the message M_1, stores everything and sends it's own data instead of client's data to the server as in $M_1^{\backslash}$.
- The server then gets the shared key K_2 with the Intruder. It then signs a message by it's private key, encrypts it with the shared key, and supposes to send M_2 to the client.
- The intruder intercepts M_2 then stores server's data and decrypts it to get the server public key then verifies the signature. Also, the intruder signs a new message using it's secret key, encrypts it with the shared key K_1 with the client, and sends it to the client.
- The client receives the message $M_2^{\backslash}$, decrypts it to get the public key from it, and validates the signature in the message for acceptance or rejection. Upon the above, the client signs, encrypts a new message M_3, and believes that it could be sent to the server.
- The intruder intercepts the message M_3, decrypts to get the public key, and validates the signature. Upon the above, the intruder can fabricate the new message $M_3^{\backslash}$ and impersonate the server by it.
- The server decrypts $M_3^{\backslash}$ and validates the signature. Upon that, it decides to accept or reject the negotiation.

From the analysis above, it is clear that the intruder can now eavesdrop, store, insert, modify, or delete all subsequent messages.
To prevent such attacks, each entity should certify the outgoing messages and verify the incoming messages. The certification must be done with all exchanges between the client and the server.

2.5 Modified STS Protocol

This section specifies and models STS protocol as stated in [13]. Modified STS-MAC protocol supposes that the second and the third messages are certified before the client decrypts it and then verifies the signature, or the server as well.

- STS-MAC protocol

- The entity A selects a random secret integer r_A and sends to B the message M_1. Upon receiving M_1, B selects a random secret integer r_B, computes the shared secret $K = (\alpha^{r_A})$, and sends message M_2 to A.
- Upon receiving M_2, A uses Cert(B) to verify the authenticity of B's signing key P_B, verifies B's signature on the message $(\alpha^{r_A}, \alpha^{r_B})$, computes the shared secret $K = (\alpha^{r_B})^{r_A}$, and verifies the MAC on $S_B(\alpha^{r_A}, \alpha^{r_B})$. A then sends message M_3 to B.
- Upon receiving M_3, B uses Cert(A) to verify the authenticity of A's signing key P_A , verifies A's signature on the message $(\alpha^{r_A}, \alpha^{r_B})$, and verifies the MAC on the $S_A(\alpha^{r_A}, \alpha^{r_B})$. If at any stage a check or verification performed by A or B fails, then that entity terminates the protocol run, and rejects.

  $M_1 : A, \alpha^{r_A}$
  $M_2 : Cert(B), \alpha^{r_B}, S_B(\alpha^{r_A}, \alpha^{r_B}), MAC_K(S_B(\alpha^{r_A}, \alpha^{r_B}))$
  $M_3 : Cert(A), S_A(\alpha^{r_A}, \alpha^{r_B}), MAC_K(S_A(\alpha^{r_A}, \alpha^{r_B}))$

  Figure 2.6 illustrates an intruder between a client and a server. once again, we study the case of man-in-middle attack. We show that the intruder can not modify the outgoing messages from the client to the server and vice versa. In addition to that it is better; the client can certify the first message instead of the third message.

- Analysis of the Modified STS protocol

  By analyzing the protocol, we find that man-in-middle attack does not have the ability to control the negotiation between the client and the server. The intruder neither shares K_1 with the client nor K_2 with the server. Also, presentation using CP-Nets, Figure 2.7, shows that the modified STS protocol is secure.

2.6 Conclusion and Discussion

As we confirmed earlier, the field of verification and analysis needs more investigation, but as time goes on, a lot of research at different universities can be done to facilitate its correctness and improve its weaknesses. The explicit Colored Petri Nets' (CP-Nets) description of the protocol entities and its attacker provides a solid foundation for protocol analysis. Other approaches used to analyze cryptographic protocols do not offer this complementary set of features.

As we mentioned, in verification and analysis tools, we chose CP-Nets as a perfect tool for our work. Our experiments showed that the insecure states could be reached by using CP-Nets.

The thesis analyzed one of the most complicated cryptographic protocols specified using CP-Nets. It added a new result for verifying and modelling STS protocol. Actually, CP-Nets helped us in proving the insecure state and how an intruder can attack STS protocol.

For the sake of simplicity, we used the concept of hierarchal CP-Nets in our analysis of the STS protocol. Moreover, in each of the steps in the hierarchy our computer program detected the insecure states. This was also evident in examining the weaknesses of the STS protocol.

2.7 Future Work

For future work, we strongly recommend more research in the area of verification and analysis of protocols.
More research needs to be conducted upon the thesis results, and we are recommending the following:

- CP-Nets should be used in the analysis of other security protocols and it is recommended to combine different cryptographic algorithms together in the analysis.
- State-space reduction methods should be developed to overcome the state-space explosion problem in the specialized CP-Nets.

Finally, several serious questions present themselves and answers to these questions should be sought. These problems are as follows:

- Is it possible to compute the performance of security protocols using CP-Nets?
- Which of the tools mentioned in Chapter 5 are most suitable for the analysis and verification of security protocols?

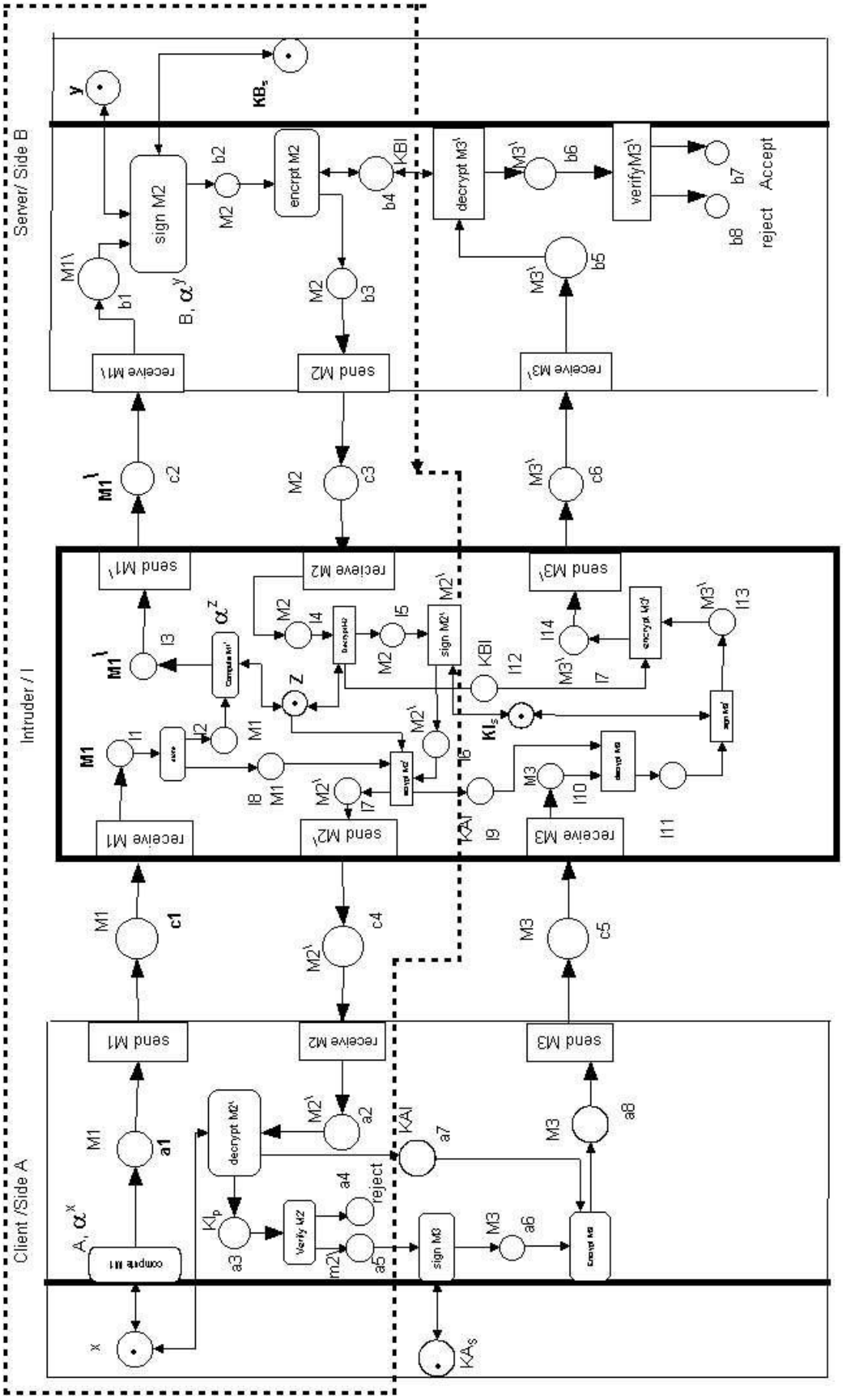

Figure 2.4: CP-Nets model for STS-intruder "insecure state part I"

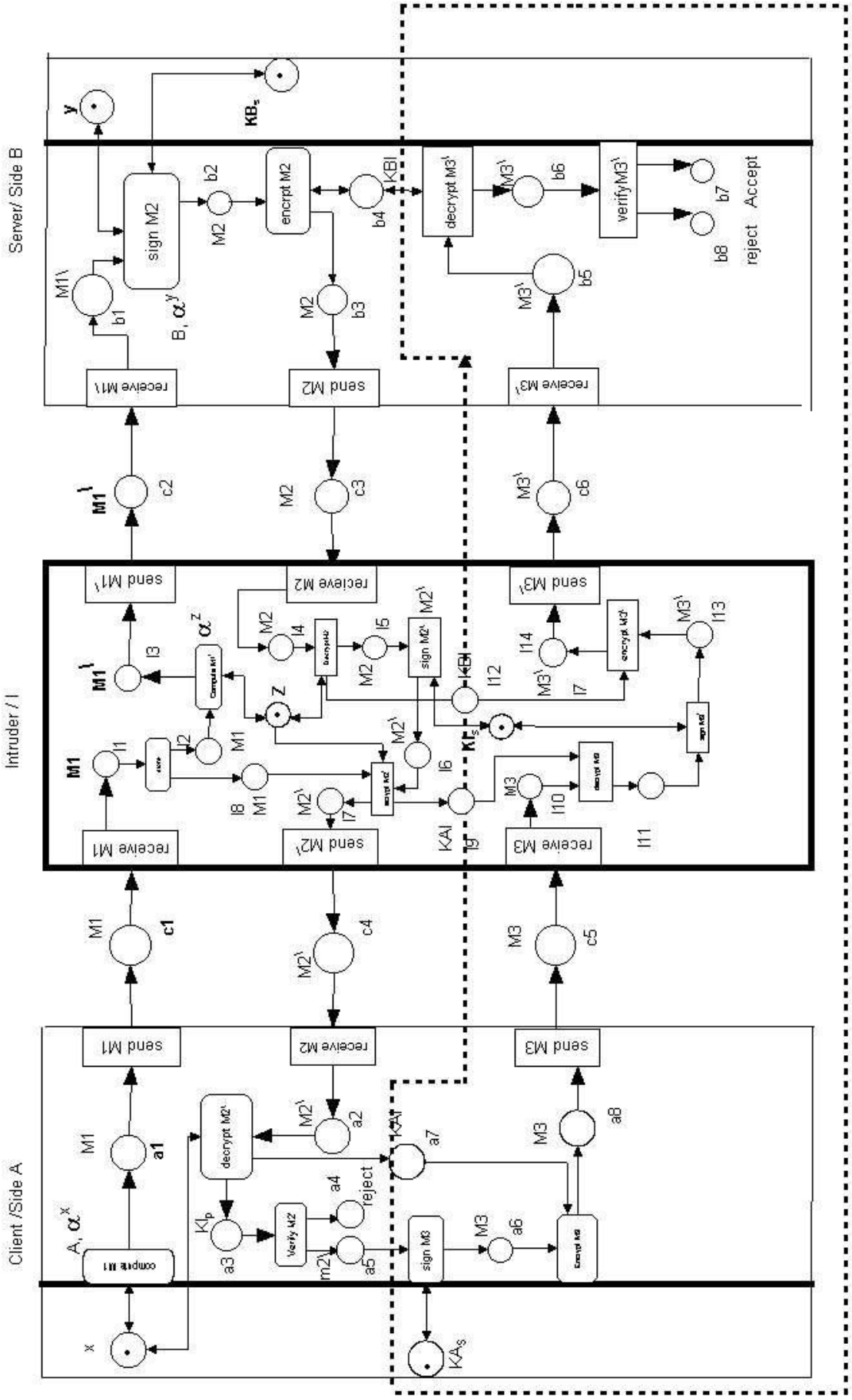

Figure 2.5: CP-Nets model for STS-intruder "insecure state part II"

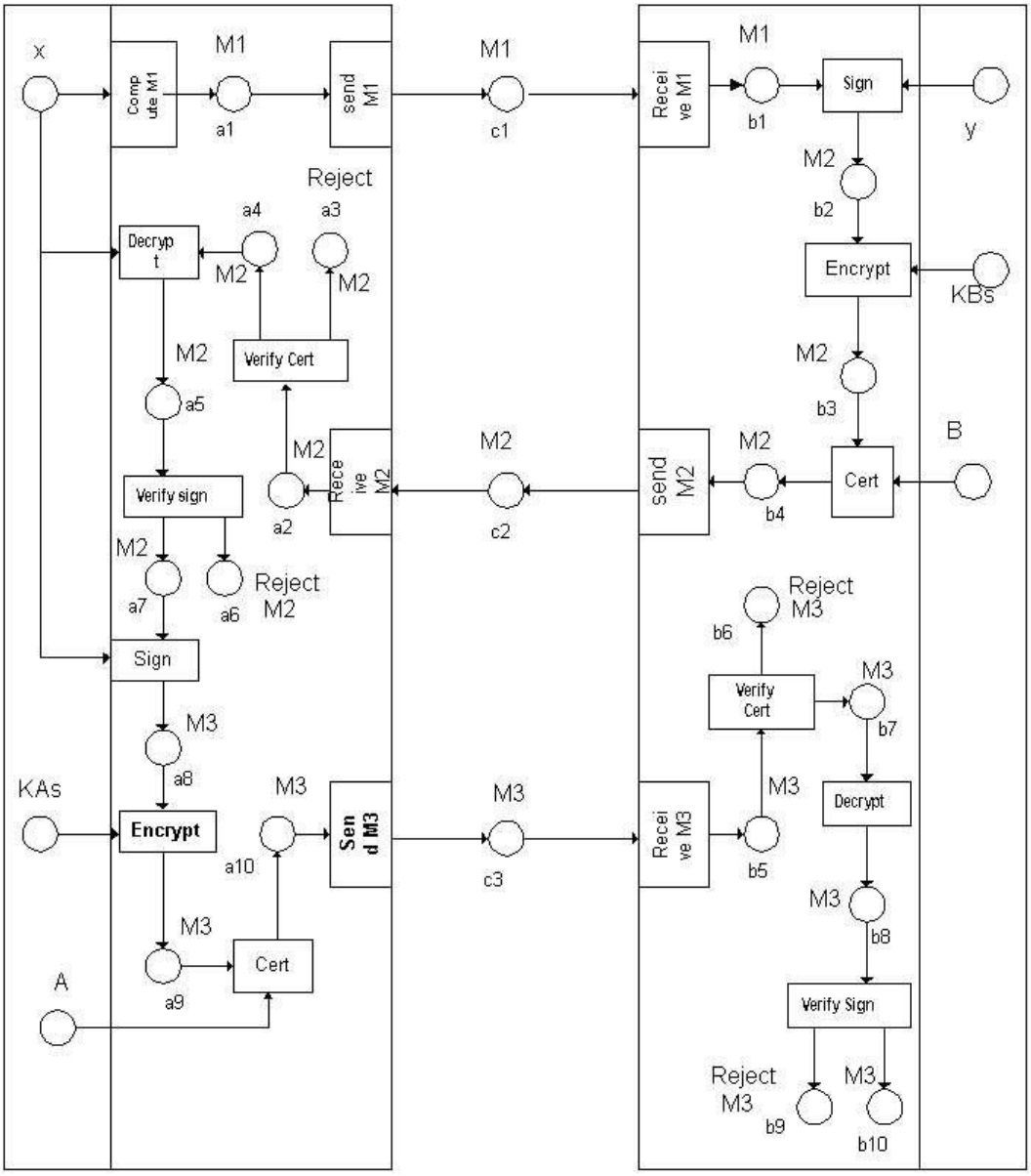

Figure 2.6: Modified STS protocol(STS-ENC-Cert)

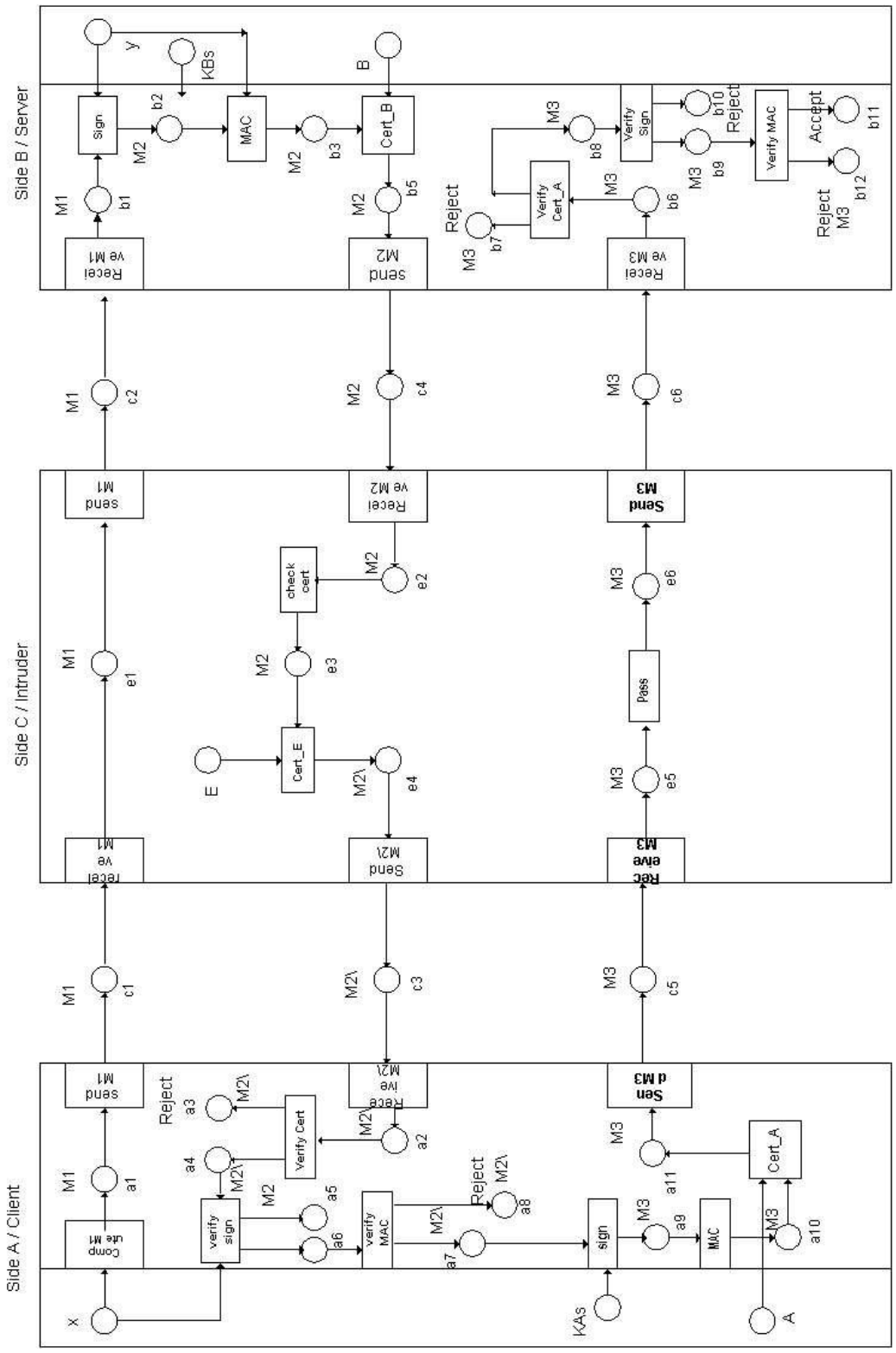

Figure 2.7: Modified STS protocol(STS-END-Cert) intruder model

Bibliography

[1] P. Ashley, M. Vandenwauver and J. Claessens, A Comparison of SESAME and SSL for Intranet and Internet Security, Information Security - Small Systems Security, Information Security Management, J.P. Eloff and R. Von Solms, Ed., IFIP Press, Vol. 2, 1998, pp. 60–69.

[2] S. Aly, Network Security Protocols: A Comparative Study, Survey and Evaluation, M. Sc., Cairo University, July, 2002.

[3] A. M. Basyouni, Analysis of Wireless Cryptographic Protocols. Master's Thesis, Queen's University Kingston, Ontario, Canada, 1997.

[4] S. M. Bellovin and M. Merritt, "Limitations of the Kerberos Authentication System," in USENIX Conference Proceedings, pp. 253–267, Winter 1991.

[5] S. Blake-Wilson and A. Menezes, "Unknown Key-Share Attacks on the Station-To-Station (STS) Protocol", Technical report CORR 98-42, University of Waterloo, 1998.

[6] G. J. Holzmann. Design and Validation Of Computer Protocols, Bell Laboratories, Prentice-Hall Englewood Cliffs, New Jersey 07632, 1991.

[7] K. Jensen, " A brief introduction to Colored Petri Nets", Workshop on the Applicability of Formal Models, 2 June 1998, Aarhus, Denmark, pages 55-58.

[8] K. Jensen. " An introduction to the Theoretical Aspects of Colored Petri Nets", Workshop on the Applicability of Formal Models, Aarhus, Denmark, 1998.

[9] 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.

[10] T. Murata. Petri Nets: Properties, Analysis and Applications. Proceedings of the IEEE, 77(4), 541-580, April 1989.

[11] L. C. Paulson. Mechanized proofs of security protocols: Needham-Schroeder with public keys. Technical Report 413, Computer Laoratory, University of Cambridge, 1997.

[12] P. Y. A. Ryan and S. A. Schneider, The Modeling and Analysis of Security Protocols, Addison-Wesley, 2001.

[13] S. G. Wasser, Mihir Bellare, "Lecture Notes on Cryptography", Cambridge Press, June 1997.

[14] World of Petri Nets, Home Page http://www.daimi.au.dk/cpnets/.

[15] Alec F. Yasinsac. Analysis of Internet security protocols, Ph.D. thesis, 1998.

	a1	a2	a3	a4	a5	c1	c2	c3	c4	b1	b2	b3	i1	i2	i3	i4	i5	i6	i7	i8
compute M1	M1																			
send M1	- M1					M1														
receive M_2																				
decrypt m2'		m2'	m2'																	
verify m2'			- m2'	m2'	m2'															
receive m1						- m1							m1							
store m1													- m1	m1						
compute m1'														- m1						
send m1'							m1'													
receive m1'							- m1'			m1'										
sign m2										- m1'	m2									
encrypt m2											- m2	m2			m2					
send m2								m2				- m2			- m2					
receive m2								- m2								m2				
decrypt m2																- m2	m2			
sign m2'																	- m2	m2"		
encrypt m2'																		- m2"	m2"	- m1
send m2"									m2"									- m2"		

Table 2.1: Modelling STS using Colored Petri Nets, step I (n=20, m=18)

	a5	a6	a7	a8	c5	c6	b4	b5	b6	b7	b8	i9	i10	i11	i12	i13	i14
sign m3	- m2'	m3															
encrypt m3		m3	KAI	- m3													
decrypt m3				m3	- m3												
receive m3					m3								- m3				
encrypt m3												KAI	m3	- m3			
sign m3'														m3		m3'	
encrypt m3'															KBI	m3'	- m3'
decrypt m3'						- m3'											m3
receive m3'						m3'		m3'									
decrypt m3'							K_{BI}	M3'	M3'								
verify m3'									m3'	- m3'	m3'						

Table 2.2: Modelling STS using Colored Petri Nets, step II (n=17, m=11)