An Attack on Digital Signature Protocol Based on Intruder's Role Impersonate*

WANG Xin-bing, MA Zheng, HUANG Lian-sheng, ZHOU Hong-bin

(*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084, *China*)

E mail: zhouhongbin99@mails.tsinghua.edu.cn

http://www.tsinghua.edu.cn

Received February 9, 2001; accepted May 25, 2001

Abstract: Security protocols play more and more important role with the bursting development of Internet. Formal authentication is an effective way to detect flaws in those protocols. There are many automatic checkers for verifying protocols, but few can deal with digital signature protocols. This paper proposes a new tool for formal automatic verifying. It can have an effective check for digital signature protocols. It is based on intruder's role impersonate and backward deduction. It was implemented by using JAVA.

Key words: security protocol; digital signature; formal authentication; intruder; role impersonate

With the rapid development of Internet, the exchange of information and communication becomes more and more frequent. The problem of security, which was not very important once, therefore, becomes dominant. Cryptic protocols are the core of security and they provide the essential authentication and encryption. Since the network service relies on the correct acts of those protocols, whether they are right or no flaw becomes the key problems.

It is hard to detect flaws of a cryptographic protocol. So people make a lot of research work. The famous method for the authentication of cryptographic protocol is the BAN logic[1] (Burrows, 1989). But without machine tools, the authentication of security must be performed by hand, whose process is very boredom and easily mistaken. So automatic authentication by computer is expected. Now there are several famous automatic authentication tools such as the Interrogator[2,3], the NRL Protocol Analyzer[2], or the use of process algebra CSP and its model checker FDR[4,5], but they all concentrate on the protocols without digital signature.

In this paper, we will introduce a new method to automatically detect the flaw in an authentication protocol, not only for the symmetric-encryption protocols and pub-encryption protocols, but also for the digital signature protocols.

* Supported by the National Natural Science Foundation of China under Grant No. 69872019 (国家自然科学基金)

WANG Xin-bing was born in 1975. His current research areas are network security, formal authentication and formal logic. **MA Zheng** was born in 1978. His current research areas are network security, applied cryptography and formal logic. **HUANG Lian-sheng** was born in 1947. He is an associate professor. His current research areas are network security, formal authentication and formal logic. **ZHOU Hong-bin** was born in 1977. His current research areas are network security, formal authentication and formal logic.

1 Method Based on Intruder's "Role Impersonate"

1.1 Introduction

M. Debbabi[6] proposed a new idea that analysis formal authentication protocol on the perspective of intruder. This idea cleverly avoid the process of formalizing the protocol to make it easier to be dealt with by computer.

In this model, protocol is described as serial messages. It introduces a corresponding "role" to each principal participating in the protocol. The role of a principal is defined as abstract of protocol step concerning the principal. Such abstract enables us to have a special comprehension on the message picked up from the exchange of protocol. An intruder answers all the challenges to impersonate the role of a principal. If it is successful, the protocol is not safe.

Through Woo and Lam protocol, we briefly introduce this method. The protocol is given below:

Step 1. $A \rightarrow B: A$

Step 2. $B \rightarrow A: N_b$

Step 3. $A \rightarrow B: \{N_b\}_{k_{as}}$

Step 4. $B \rightarrow S: \{A, \{N_b\}_{k_{as}}\}_{k_{bs}}$

Step 5. $S \rightarrow B: \{N_b\}_{k_{bs}}$

First we extract the roles of the protocol:

$\mathrm{Role}(A) = \langle O, A, B\rangle \langle I, N_b, B\rangle \langle O, \{N_b\}_{k_{as}}, B\rangle$

$\mathrm{Role}(B) = \langle I, A, A\rangle$
$\langle O, N_b, A\rangle$
$\langle I, \{N_b\}_{k_{as}}, A\rangle$
$\langle O, \{A, \{N_b\}_{k_{as}}\}_{k_{bs}}, S\rangle$

$\mathrm{Role}(S) = \langle I, \{A, \{N_b\}_{k_{as}}\}_{k_{bs}}, B\rangle$
$\langle O, \{N_b\}_{k_{bs}}, B\rangle$

The role "A" has three triples, which represent three steps it participates. In each triple, if the first part is "O", "A" is the message sender and if is "I", "A" is the message receiver. The second part represents the message exchanged in the protocol. The third part represents the other communicating principal. So do role B and S.

Then we generate verification system. When analyzing the security protocol, we always posit the existent of some intruder. He can intercept, alter, memorize, index and utter any information flowing in the network. He also has the general capacity of encryption and decryption. In the method of "role impersonate", an intruder has another valuable source of information: the protocol. An intruder can view a protocol as a computing facility, utilized to generate and synthesize some particular information. In some case, a lethal attack can be deduced. Such capacity of attack is realized in the verification system including reasoning-rules.

Each reasoning-rule in the verification system is corresponding to a step of a protocol. It has the following form:

$$\frac{p_1 \ldots p_n}{m}_c$$

in which, "p_i"$(i=1\ldots n)$ and "m" are messages, while "c" is a Boolean formulation. This rule denotes that when "c" is true, an intruder can get "m" from the protocol, only if he provides p_i to the protocol. More over, when each rule is given, we can get the detail steps how an intruder deduced "m" from "p_i", i.e., attack scenario. In practice, "c" is freshness in general.

In the scenario below, the attack of an intruder is demonstrated.

a_1 $I(A) \rightarrow B: A$

a_2 $B\rightarrow I(A):N_b$

a_3 $I(A)\rightarrow B:anyvalue$

a_4 $B\rightarrow I(S):\{A,anyvalue\}_{k_{bs}}$

b_1 $C\rightarrow I(D):C$

b_2 $I(D)\rightarrow C:N_b$

b_3 $C\rightarrow I(D):\{N_b\}_{k_{cs}}$

c_1 $I(C)\rightarrow B:C$

c_2 $B\rightarrow I(C):N_b$

c_3 $I(C)\rightarrow B:\{N_b\}_{k_{cs}}$

c_4 $B\rightarrow S:\{C,\{N_b\}_{k_{cs}}\}_{k_{bs}}$

c_5 $S\rightarrow I(B):\{N_b\}_{k_{bs}}$

a_5 $I(S)\rightarrow B:\{N_b\}_{k_{bs}}$

where a, b, c represent three instances of the protocol respectively. At a_3, the intruder "I" successfully impersonates "A" to principal "B". It is clear that between c_3 and c_5, the intruder utilizes authentication-center "S" as the Oracle to get $\{N_b\}_{k_{bs}}$, a challenge that principal "B" puts forward in the protocol instance a.

2 Authentication Tool Based on Intruder

2.1 Methodology

After analysis and compare, we decided to adopt the formal analysis of security protocol based on intruder. There are a lot of restricts, when using Murϕ[7]. Stanford University and CMU both have hardware and software about finite-state system. From the result of their experiment, we can get the conclusion that space-time complexity will bulge to exponential grade with the increase of protocol sponsor, protocol receiver and protocol attacker. In such situation, it is very difficult to make real sense in implementing the authentication tool. While to the logic tools of BAN logic, it is the bottleneck to formalize the protocol. On the contrary, in the view of efficiency and formalization, analysis algorithm of security protocol based on intruder has the characters of polynomial grade with the increase of communicating principals, steps of the protocol and the number of intruders. Only simple formalizations to yield the roles corresponding to each principal respectively are needed, which can be done entirely by computer program. So we believe that the analysis algorithm of security protocol based on intruder is an appropriate method.

2.2 Tools

We choose JAVA as our tool, for JAVA is the popular advanced language, which has the parallel function and is completely object-oriented. Without any revise, JAVA can run on different operating system.

2.3 Brief introduction to the algorithm

We propose three suppositions based on the algorithm we will introduce.

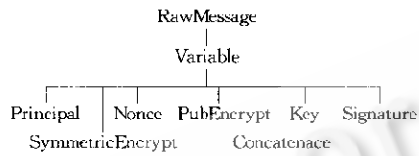
1) The network on which security protocols run is open. The message any principle sends can be received, altered, discarded and substituted by any other principle.

2) In the running process of security protocol, an intruder is a legal principal. He can provoke and participate in any instance of the running protocol. In other words, an intruder can completely use the computation of the protocol itself.

3) The messages transmitted in the running process of security protocol are nonstructural and not dealt with Hash function and signature. That is, a principal cannot discern any message such as Nonce, Key from district of messages.

Authentication tool is realized with JAVA. For JAVA is purely object-oriented, it is necessary to define the classes for all concepts, and encapsulate the core algorithm as method for the proper class.

First we define the most basic message-base class: RawMessage. All the message classes transmitted on the network are derived from Variable class, which is the son-class of RawMessage. The message classes are Principal class, Nonce class, PubEncrpt class, SymmetricEncrypt class, Concatenance class , Key class and Signature class. The relationship between these seven classes and RawMessage class is shown below.



The core class is Deduction class, whose method of Deduction is the core authentic algorithm. The Deduction class has two attributes: m_Open and m_Close. m_Open represents the formulations to be verified and m_Close represents the formulations having been known or verified. If m_Open is empty, then we get an attack on that protocol.

3 Attack upon Digital Signature Protocol

We have already verified some protocols using the tool we developed, such as Needham Schroeder protocols[8]. They are all protocols without Digital Signature, for a protocol using digital signature has very high quality of security. Now the tools for automatic authentication always aim at protocols without digital signature.

We improve our tool to verify some digital signature protocols.

For example, the station-to-station protocol[9,10]

The station-to-station protocol consists of Diffie-Hellman key establishment followed by mutual authentication. The key establishment assumes the existence of a publicly-known cyclic group and a corresponding primitive element a.

Step 1. $A \rightarrow B: A, B, a^x$

Step 2. $B \rightarrow A: B, A, a^y, \{S_B(a^x, a^y)\}_k$

Step 3. $A \rightarrow B: A, B, \{S_A(a^x, a^y)\}_k$

where $k=a^{xy}$, S_A is A's signature and S_B is B's signature.

First, we give the machine description of the protocol.

Principals:

$A(?)$—NORMAL PRINCIPLE

$B(?)$—NORMAL PRINCIPLE

$I[A(?)]$—ATTACKER

Messages:

$A(?) \rightarrow B(?): A(?), B(?), a^x$

$B(?) \rightarrow A(?): B(?), A(?), a^y, \{(a^x, a^y)\_k(sign)(B(?), B(?))\}\_k(pri)(A(?), B(?))$

$A(?) \rightarrow B(?): A(?), B(?), \{(a^x, a^y)\ k(sign)(A(?), A(?))\}\ k(pri)(A(?), B(?))$

Secrets after the protocol:

$k(pri)(A(?), B(?))$

Then, generate rule-templates according to the protocol itself. These templates are outputted below:

Deducing rules:

RuleGenerator:

To Impersonate $A(?)$

premises:

$A(?)$, $B(?)$, $MSG(?)$

SideConditions:

Conclusion: $B(?),A(?),MSG(?),\{(MSG(?),MSG(?))\_k(sign)(B(?),B(?))\}\_k(pri)(A(?),B(?))$

Variable used:

$MSG(?),A(?),B(?)$

RuleGenerator:

To Impersonate $B(?)$

Premises:

SideConditions:

Conclusion: $A(?),B(?),MSG(?)$

Variables used:

$MSG(?),A(?)$

RuleGenerator:

To Impersonate $B(?)$

Premises:

$A(?),B(?),MSG(?)$

$B(?),A(?),MSG(?),\{(MSG(?),MSG(?))\_k(sign)(B(?),B(?))\}\_k(pri)(A(?),B(?))$

SideConditions:

Conclusion: $A(?),B(?),\{(MSG(?),MSG(?))\_k(sign)(A(?),A(?))\}\_k(pri)(A(?),B(?))$

Variables used:

$MSG(?),A(?)$

Then, we get $m_$Open set and $m_$Closed set below:

To Impersonnate $B(?)$

Open Set: $\{(MSG(?),MSG(?))\_k(sign)(B(?),B(?))\}\_k(pri)(A(?),B(?))$

Close Set: $k(sign)(I,I)$

$A(?)$

$B(?)$

$I(?)$

After the prepared works above, the program generates reasoning-rules and axioms according to the protocol, and deduces the knowledge of $m_$Closed and $m_$Open. In the end, we get the empty of m Open set as a result. It means there are some flaws in the protocol. In fact we can get the attacking scenario from the deduction.

Step 1. $A\rightarrow I(B):A,B,a^x$

Step 2. $I\rightarrow B:I,B,a^x$

Step 3. $B\rightarrow I:B,I,a^y,\{S_B(a^x,a^y)\}_k$

Step 4. $I(B)\rightarrow A:B,A,a^y,\{S_B(a^x,a^y)\}_k$

Step 5. $A\rightarrow I(B):A,B,\{S_A(a^x,a^y)\}_k$

4 Conclusions

During recent research, we found that little work had been done in the field of formal authentication about digital signature protocol. There are several reasons. It is hard to find a suitable protocol to verify, most of digital signature protocol have high levels of security. Another reason is that it is hard to express the structure of digital signature. So many authentication tools, such as NRL, FDR, choose symmetric-encryption protocols and pub-encryption protocols as their targets. In this paper, we regard digital signature as pub-encryption and eliminate the detail properties inside. So we can detect the flaws on structure of digital signature protocols.

How to express the properties inside of digital signature protocols and how to detect those flaws are the further research work.

References:

[1] Burrows, M., Martin, Abadi, Roger, Needham. A logic of authentication. ACM Transactions in Computer Systems, 1990,8(1):18~36.

[2] Kemmerer, R., Meadows, C., Millen, J. Three systems for cryptographic protocol analysis. Journal of Cryptology, 1994,7(2):79~130.

[3] Millen, J.K., Clark, S.C., Freedman, S.B. The interrogator: protocol security analysis. IEEE Transactions on Software Engineering, 1987,13(2).:274~288.

[4] Lowe, G. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In: Margaria, Steffen, eds. Proceedings of the TACAS. Volume 1055 of Lecture Notes in Computer Science, Springer-Verlag, 1996. 147~166.

[5] Roscoe, A.W. Modelling and verifying key-exchange protocols. In: Proceedings of the 9th IEEE Computer Security Foundations Workshop. IEEE Press, 1995. 98~107.

[6] Woo, T.Y.C., Lam, S.S. Authentication for distributed systems. Computer, 1992,25(1):39--52.

[7] Mitchell, J.C., Mark, Mitchell, Ulrich, Stern. Automated Analysis of Cryptographic Protocols Using MurΦ. In: Proceedings of the 1997 IEEE Symposium on Security and Privacy. IEEE Computer Society Press, 1997. 141~151. http://www.computer.org/proceedings/7828toc.htm.

[8] Huang, Lian-sheng, Wang, Xin-bing, Xie, Feng, *et al*. Formal authentication based on intruder's role impersonate. Journal of Tsinghua University, 2001,41(7):72~75 (in Chinese).

[9] Diffie, W., van Oorschot, P.C., Wiener, M.J. Authentication and authenticated key exchanges. Designs, Codes and Cryptography, 1992,2:107~125.

[10] Gavin, Lowe. Some new attacks upon security protocols. In: Proceedings of the 1997 IEEE Symposium on Security and Privacy. IEEE Computer Society Press, 1997. 141~151. http://www.computer.org/proceedings/7828toc.htm.

附中文参考文献:

[8] 黄连生,王新兵,谢锋,等.基于攻击者的"角色冒充"的协议验证方法.清华大学学报,2001,41(7):72~75.

基于角色冒充对数字签名协议的攻击

王新兵, 马 征, 黄连生, 周宏斌

(清华大学 计算机科学与技术系,北京 100084)

摘要: 实现对具有数字签名的身份认证协议进行验证.从攻击者的角度,基于角色冒充的方法,使用逆向归结的推理方式,采用JAVA来实现验证算法.对于一些具有数字签名的协议,可以进行有效的验证.

关键词: 安全协议;数字签名;形式化验证;攻击者;角色冒充

中图法分类号: TP309 **文献标识码:** A