Regular Paper

A Study of Access Control Method for Mobile Agents by Using Secure Stubs

Shinsaku Kiyomoto,† Toshiaki Tanaka† and Koji Nakao†

There has recently been active research on mobile agent technology. Very few, however, address security issues for a practical service model such that a number of service providers are engaged over the heterogeneous network. In this paper, we present a new approach on "access control" to realize global authorization under such environments. In our method, policy files and information related with authentication for an agent are consistently managed and efficiently processed by an authentication server of each network. Furthermore, for the purpose to enhance the security and realize the efficient access control, new interface modules called "secure stub" are proposed. As the result of its evaluation, it can be basically feasible to operate in our proposed system architecture. We believe that the work presented here is significant to the advancement of the existing mobile agent environments supporting secure communications.

1. Introduction

Recently, a software agent technology is highly paid attention to achieve sophisticated services such as electronic shopping and online reservations over the Internet and/or Mobile networks. A software agent may be defined as a set of programs that autonomously acts on behalf of a user. If the agent has capability of migrating from home to foreign networks, or from foreign to foreign networks, then it is specifically called as "Mobile Agent". In the above situation, to guarantee agents mobility among non-trusted networks, various types of threats on security are to be considered such as masquerading or damaged by malicious agents, exposure of a sensitive information by eavesdropping and so on [1]. It is required to provide security countermeasures, especially for authentication and access control of the mobile agents. This paper, therefore, discusses about " authentication and access control mechanism" to satisfy security requirements under the practical environment. We have introduced a centralized access control mechanism by each network, so that an authentication server will authorize all the agents registered to it, wherever they go. Furthermore, for the purpose of enhancement of the security and realization of the efficient access control, interface modules called "secure stub" are newly proposed. The centralized approach also solves the open issue on multiple-hop authentication. We have evaluated it from the point of a program size of it and computational time to show its feasibility. The rest of the paper is organized as follows. Related works are introduced in Section 2. A basic concept of our proposed method shows in Section 3. Section 4 provides more detailed proposal of our method including security assumptions and mechanisms. Section 5 gives performance evaluation of the prototype system. Section 6 introduces considerations on our research. Finally, we conclude our paper in Section 7.

2. Related Work

Authentication and access controls of foreign agents are key technologies for the purpose to protect hosts where the agents are executed. Up to now, Jansen [4] proposed access control mechanism by using privilege certificates, which are issued by a service provider and defines privileges consisting of a list of allowable services for a user. In this method, an agent migrates with his certificates, and hosts can control access of the agent according to privileges written in the certificates. This method can be applied to the mobile agent environment. The simplest system utilized by the mobile agent technology may consist of one service provider over a number of networks. The user makes use of its services according to primitives allowed by the service provider. As a practical case, however, the system may be constructed by a number of service providers. Then, a mobile agent may migrate from one to the other network conveying his all privilege certificates. In the case, however, some privacy information contained in his

† KDDI R & D Laboratories Inc.

privilege certificates may be exposed to non-trusted network. The management of the access control that depends on each privilege certificate is rather complicated. With regard to authentication, digital signature technology can be applied as one solution. However, authentication of multiple hop migration still remains as an open issue. To solve the above issue, a host may dynamically authenticate a foreign agent when it migrates. In the case, the agent should convince the host that the agent is valid by using his private information such as private key or shared key. Therefore, strength of authentication depends on how private information is securely stored within the agent to protect the agent against eavesdroppers or malicious host.

A major technology protecting mobile agents is tamper resistant techniques of agent programs, such as code obfuscation and mobile cryptography [2]. Mobile cryptography is a method to request a mount of computation for a remote agent without knowing the value of the result of the computation directly. It seems to be an ideal solution because its technique is provably secure from the point of the modern cryptography. However, it can be used for very limited functions such as homomorphic functions.

Code obfuscation is a practical solution to hide secret information within the code. The secrecy, however, is not hold permanently because its security is not based on the modern cryptography. Hohl proposed a concept of code obfuscating techniques called *Time Limited Blackbox Protection* [3]. An adversary can analyze programs, and expose secret information in programs. But, if the lifetime of a program is enough shorter than the analysis time of the adversary, it cannot threaten the security of services. Because the secret information in the program goes to waste after the lifetime. He also introduced some mess-up algorithms realizing the above concept, which make program codes hard to analyze. For example, *Variable Recomposition* algorithm cuts each variables into segments and creates new variables that contain a recomposition of the original segments, and *Conversion of compile — time control flow elements into run — time data dependent jumps* algorithm translates *if* and *while* statements into sets of many statements having many variables. The strength of these algorithms may depend on complexity or computational costs of the algorithms, however, it can not determine quantitatively because of the current lack of a formal model of the agent mess-up and the associated counter algorithms.

Wang, et al. proposed a code obfuscation based on transformation using pointer manipulations, which was proven theoretical secrecy [5]. They showed that statically determining precise indirect branch addresses was a NP-complete problem in the general pointers. However, Wang's method has not been proven secrecy against practical attacks or dynamic attacks such as insertion of test codes and testing.

In the following section, we discuss the problems of Jansen's mechanism and propose our method to solve these problems.

3. Basic Concept of Our Method

3.1 Existing Method of Access Control

An existing method of access control proposed by Jansen will be modeled in **Fig. 1**. In the existing method, each platform has his policy files and an agent has privilege certificates. The privilege certificate of the agent contains privileges of its owner or a list of permitted actions for each service, and is attached to the agent when the agent is invoked. The agent migrates from a home (registered) platform to a foreign (non-registered) platform, and provides his privilege certificate to a foreign policy engine. The policy engine is one of the additional modules implemented in the server, which generates a security policy and provide a decision to permit or deny agent's actions according to the security policy. The security policy consists of allowable privileges for the agent, representing the amalgamation of policy rules within the privilege certificate of the agent and the policy file of the platform. The privilege certificate should be authenticated by each platform.

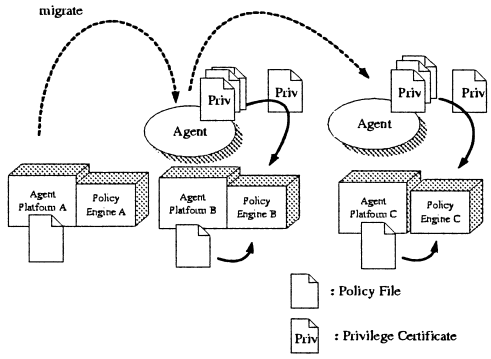

Fig. 1 Overview of existing method.

The existing method seems to be one of the reasonable approaches. However, the following problems can be identified when applying the existing method to the agent system.

- If the migrating platform is malicious one, then information of owner's privileges will be disclosed to the platform when the agent provides it to the platform. This might be a critical problem in the agent system.
- An authentication of a foreign agent is one of difficult problems in a mobile agent environment. One of popular method is that an agent is digitally signed to support message origin authentication, and a platform checks the signature. However, we have to take into consideration that the status of the mobile agent may change with the result of execution, while the agent migrates among two or more platforms. Therefore, the authentication by using a digital signature is not preferable solution under the multiple hop environments.
- An agent has to convey privilege certificates of all services for using the agent in every migrations. In this case, the total size of these privilege certificates are probably large. This might be a disadvantage in view of performance. Further, all platform has to implement the policy engine which is additional module.

3.2 Basic Concept of Our Method

Our proposal is a new approach to control accesses with the following characteristics as shown in **Fig. 2**.

- User privileges are centralized managed by a Policy file Management Module (PMM) of each network. The privileges are described in a user's policy file.
- When an agent migrates to a foreign platform, the agent shall have access to the PMM to which the agent belongs in order to obtain the permission of access.
- Eventually, it is not necessary for the agent to convey its privilege certificates when migrating to the foreign platform.

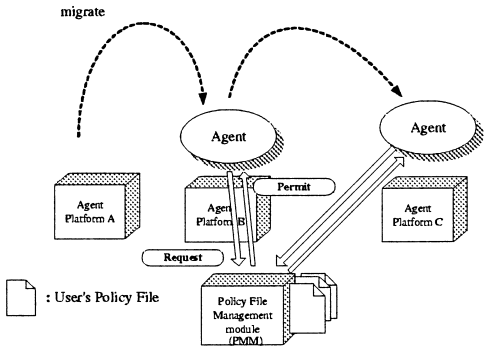

Fig. 2 Overview of proposed method.

Our proposed method solves the previous problems in the following ways.

- All user's policy files are securely managed by the PMM. An agent only inquires whether the requested access is permitted or not by the PMM. It means that needless information of user's privilege does not leak to a malicious platform, and a platform need not to confirm that a privilege certificate is securely attached to the agent.
- An agent is authenticated by the home PMM of the agent, a platform need not to authenticate foreign agents.
- An agent does not convey privilege certificates, then the size of agent is probably small, comparing with the existing method.

4. Detail Mechanism Based on Our Proposed Method

4.1 Basic Components of Our Proposal

We assume that a mobile agent system consist of the following three kinds of computers.

- User terminal
  A user terminal is a front-end computer, where a human user requests tasks and gets their results. It is directly facing to an authentication server.
- Authentication server
  An authentication server provides an intermediate function between the user terminal and agent activating servers and also manages a lifecycle from creating of agents to deleting. An authentication server manages any information related to authentication of agents and users, which belong to his domain. The PMM is implemented in this server.
- Agent activating server
  An agent activating server is a playground of an agent to work out the requested tasks on behalf of a user.

4.2 Authentication and Access Control Mechanism of Our Method

The PMM is implemented in the authentication server, and manages an user's policy file. Furthermore, a secure stub which controls an agent to have access to service resources, is in-

troduced in this method. The secure stub is initially registered by the service, which is located on the agent activating server to provide mediation between the agent and the service. The agent cannot do any action on an agent middleware without stubs. The process of access control based on our method is the following.

(1) *Invoke* : An agent is invoked by an authentication server, according to user's request.

(2) *Migration* : The agent migrates from the home platform to the agent activating server.

(3) *Authentication* : The agent requests using a service to the authentication server, and the server authenticates the agent.

(4) *Authorization* : The authentication server authorizes the agent by using the policy file of its owner, which is already stored.

(5) *Send a secure stub* : If the agent has the privilege of using the service, the authentication server sends the secure stub of the service to the agent.

(6) *Use a service* : The agent uses the service through the secure stub.

4.3 Security Assumptions in Our Method

The basic assumptions of our method are as for the following.

- Authentication server
  The authentication server is assumed to be a trusted party that is protected against network attacks and computer viruses. In the server, user information and other classified information such as user's policy files are securely managed and all of the processes are honestly worked.
- Agent
  The agent is assumed to be tamper resistant software by using tamper-resistant techniques such as Hohl's method in Section 2.
- Services
  Services are programs produced by service providers and registered by a manager of the agent activating server. In this case, the manager can verify origin of the service and validity of the service before the service is in use. Therefore, it can be assumed that services are honest programs.
- Secure stub
  We assume that the stub is also tamper resistant software. It is implemented by using tamper-resistant technique as in the case of agents. The stub is registered to the authentication server by the service provider on a secure cannel.
- Security of encryption/decryption function
  It is assumed that the encryption f and decryption f^{-1} functions are secure against well-known attacks.

4.4 Detail Mechanism

The protocols of our method are as for the following.

(1) First, the agent sends his name and the service names to the authentication server.

(2) The authentication server authenticates the agent by using $IDag$ and refers to the policy file of the owner of this agent. When the agent is invoked, information of the agent that is $IDag$, is uniquely assigned and securely embedded in the agent by the authentication server. The process of authentication is as for the following. Where, | indicates the concatenation of the data, R, R' are random numbers, Sv indicates the authentication server, Ag indicates the agent, $\rightarrow$ indicates a data flow between two entities, and $\stackrel{?}{=}$ indicates a comparison whether left side is equal to right side.

$$Sv \rightarrow Ag : R$$
$$Sv \leftarrow Ag : f_{IDag}(R \,|\, R')$$
$$Sv : R \stackrel{?}{=} R \text{ from } f^{-1}_{IDag}(f_{IDag}(R \,|\, R'))$$

(3) The authentication server check the policy file of the owner by service names which sent by the agent. If the agent has privilege to use the service, the authentication server has securely embedded R' and install its privilege into the secure stub of the service, which is created by each service provider. The authentication server adds the digital signature to this stub and its expiration date, and replies to the agent.

(4) The agent confirms that the secure stub was created by the trusted authentication server, as in the following process. Where, R'' is a random number, and St indicates the secure stub.

$$Ag \rightarrow St : R''$$
$$Ag \leftarrow St : f_{R'}(R'')$$
$$Ag : f_{R'}(R'') \stackrel{?}{=} f_{R'}(R'') \text{ from } St$$

(5) The agent has access to the service through the secure stub. The process of access is as for the following. The agent encrypts messages by using R', and sends this message to the secure stub. The secure stub decrypts the encrypted message by R', transforms from the decrypted message to the service understandable

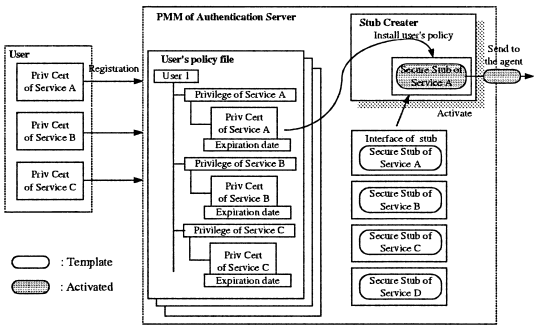

Fig. 3 Creation of secure stubs.

form, and sends to the service. The response that the message is replied to the agent is the contrary protocols.

4.5 Creation of User's Policy Files and Secure Stubs

This section describes how the authentication server creates user's policy file and secure stub as follows (**Fig. 3**). At first, PMM creates user's policy file when a user registers to the authentication server. User's policy file is also updated when a new privilege certificate is registered by the user. The policy file is defined as a set of privileges based on privilege certificates, and it contains the privilege certificate. Therefore, when a privilege certificate is expired, the PMM deletes it on the policy file and reconstruct the policy file. Next, service provider prepares template of the secure stub (the template hereafter) and its interface module. He creates the template not to access to critical resource on the authentication server and digitally signs with his private key. Then he authenticates the authentication server and registers them to the server. At the registration, PMM verifies the signature of them. If verification is success, PMM stores them.

At the authentication phase, the PMM checks whether a name of requested service exists or not. If the name exists, the PMM drives creation process of a secure stub. At first, the PMM finds a privilege certificate of the service and a template of secure stub of the service. Then he installs the certificate to the interface of the template. The interface interprets the certificates, and extracts information of detailed privileges from its certificate. Then the privilege information is converted into a specific form such that the template can easily recognize, i.e., binary form (11011011$\cdots$). A random number R' sent by the agent is also input to the template by the interface. When the creation process is finished, the PMM digitally signs the template as an (activated) secure stub and sends back it to the agent. The secure stub controls accesses to service resources by means of the above privilege information in a specific form.

5. Performance Results

In order to evaluate the system performance regarding management of the secure stub, the following issues are significant points to be measured:

- Process burden to authenticate an agent and to generate a secure stub at the authentication server.
- Communication burden to authenticate and authorize the agent.
  This depends on program size of the stub, because the size of the other communicating data such as authentication token is quite small.
- Process burdens at the agent activating server, i.e.,
  - Burden of authenticity check of the stub by the agent.
  - Total throughout of the stub from the agent to the service.
- Capacity or Scalability of the authentication server.

We implement and evaluate the authentication server and the agent activating server by using personal computers (PentiumIII 800 MHz). A existing obfuscator and a segmentation algorithm of secret key such as *Variable Recomposition* are used to obfuscate an agent and a secure stub. A time of downloading is including authentication process of an agent. The cipher algorithm is a 64 bit block cipher. In this evaluation, Java common libraries are used at the stub, are those provided in JDK 1.3, i.e., java.security, java.util, and javax.crypto packages.

According to the performance evaluation results in **Table 1**, the secure stub can be basically feasible to operate in our proposed system architecture. First, the size of stub program is small enough to download from the authentication server. Second, burden for the generation process of the stub is also totally negligible. Third, the total throughput is feasible for agent services, approximately 3.1 Mbps. Therefore, there is no specific problem to realize the secure stub except the burden of the authenticity check of the stub at the authentication

Table 1 Result of measuring the secure stub.

Stub program size	5.42 KB
Java common libraries	43.93 KB
Total processing time of authentication and authorization	390 msec
Authenticity check	1,167 msec
Total throughput from an agent to a service	3.1 Mbps

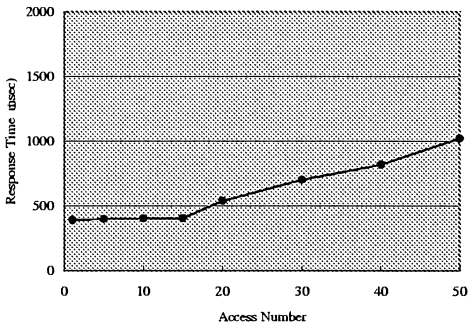

Fig. 4 Response time of authentication and authorization process.

server.

In the next step, we evaluate capacity of the authentication server. We measure response time of authentication and authorization process, when many agents are simultaneously requesting the server. The average result of 200 times evaluation is illustrated in **Fig. 4**. When the transaction numbers are less than 15, the processing times are all about 390 msec. When the numbers are more than 20, the processing time is linearly increasing by the number, and the time is about 1,000 msec at 50 transactions. The capacity of the authentication server is therefore, about 50 transactions/sec. In common Web sites, a number of access is 10–20 transactions per second. In mobile agent environments, the number will be less than that of Web sites. Therefore, the authentication server has sufficient capacity for the environment.

6. Considerations

6.1 Security Considerations

In this section, we discuss security of the proposed mechanisms.

- Masquerading by a malicious agent
  The authentication server authenticates the agent by using secret information, i.e., $IDag$. When the agent is invoked, $IDag$ is embedded by the authentication server on a secure channel. Moreover, the secrecy of $IDag$ is assured because the authentication server is trusted and the agent is tamper resistant.
- Masquerading by a malicious authentication server
  In this mechanism, an authentication server is not directly authenticated by an agent. An agent, however, authenticates it indirectly by confirming that the downloaded secure stub can compute $f_{R'}(R'')$ correctly. Because, a malicious authentication server cannot decrypt $f_{IDag}(R|R')$ correctly, therefore, the server cannot create a correct secure stub.
- Unauthorized access by an agent: invalid downloading
  The authentication server, that is a trusted party, securely manages the policy of the users and the secure stubs. A secure stub corresponding to a permitted service can be downloaded by the agent only when he is correctly authenticated and authorized by the authentication server.
- Unauthorized access by an malicious agent: re-use of a secure stub
  Even if a malicious agent steals a secure stub for its re-use, the malicious agent cannot use the interface provided by the stub, because he does not have the knowledge of a secret key R', which is recognized at the authentication procedure. Further, expired stubs can not be used.
- Unauthorized access by a malicious agent: forgery of a secure stub
  In this mechanism, a secure stub is digitally signed by the authentication server. Therefore, the alteration of secure stub is impossible.
- Invalid access to agent information by a malicious secure stub
  Even if a malicious authentication server forges a secure stub, an agent can distinguish it from a valid secure stub, because R' in the agent is not identical to R' in the stub.
- Denial of service attack by a malicious agent
  The malicious agent cannot use secure stubs, therefore, he cannot try any action on the platform. Therefore, the success probabilities of the above malicious attacks are quite negligible. Even in this mechanism, it is possible for an authenticated agent to randomly request services in or-

der to damage the system resources in the platform. The protecting method against the threat is for further study.

- Term of validity for agents and secure stubs
  Term of validity for agents and secure stubs depend on strength of tamper-resistant techniques. Currently, there exists no general method to evaluate the strength of software tamper-resistant techniques, because of the lack of a formal model of code obfuscation and the other algorithms. Generally, a cost of analysis for obfuscated code depends on complexity, such as complexity of variables, complexity of statements, complexity of branches, and complexity of dummy statements. The more code is complex, the more the code is secure.
  As the drawback, the cost of execution is increased. Namely, it is trade-off between the cost of execution and the strength of obfuscation. Therefore, the cost of obfuscation should be decided according to lifetime of a program and security requirement. For example, obfuscation of the secure stub can be simpler than that of the agent, because lifetime of the stub is shorter. A margin between the cost of obfuscation process and the cost of its analysis depends on the advantage of knowledge of an original source code and its obfuscation algorithms against adversaries. If the code and all algorithms of obfuscation are published, the analysis may be easy.
  Matsuoka [6] proposed evaluation results of code obfuscation by human evaluators. In the results, a skillful evaluator can expose secret information from an obfuscated program in 7.5 – 43 hours by using the original source code and the obfuscated program. We use similar obfuscating techniques, i.e., segmentation and permutation of variables. Furthermore, we conceal original source codes against an adversary. Therefore, it is expected that the term of validity of our obfuscation is a few hours at least. The term is feasible for mobile agent services.

6.2 Comparison with the Existing Method

In our method, privilege certificates and user's policy files are centrally managed on the PMM of each network. Our architecture exhibits resemblance to traditional secure distributed system other than to ordinary agent architectures. In the existing method, mobile agent migrates with his privilege certificate. In the case, some privacy information contained in his privilege certificates may be exposed to non-trusted network and platform. In our method, the above concern needs not to be taken into consideration, because it manages privilege certificate within the trusted PMM. Our method also solves an open problem of multiple-hop authentication of the agent, because the authentication server has secret information sharing with agent and the server can easily authenticate the agent under the condition that the tamper resistant assumption holds. Our method needs not to implement additional function to interpret and amalgamate policy files such as a policy engine on all agent platforms. Nevertheless, well-known advantage of the existing method such as complete offlineness of agents or scalability should be carefully taken into account. Our method needs to access the authentication server every time a mobile agent migrates. Therefore, communication and computation burden of the authentication server is critical. However, the evaluation result in Section 4 shows that our system is feasible against the burden applying to the practical network scale. Now, we define platform number of migration of a agent is N, data size of a privilege certificate which is used in existing method is p, data size of a secure stub is s, total data size of authentication process in our method is c, and program size of an agent after calculation on platform i is A_i. Communication burden of all migration is defined as the following, where the agent uses different service on every platform. In the existing method, an agent migrates $N + 1$ times with N privilege certificates for each platform. We assume that a size of privilege certificate is about 1 KByte in the existing method. Our method is more efficient compared with the existing method, where $N > 5$. That is to say, our method is efficient where the agent migrates many platforms in range of agent's lifetime restricted by strength of a tamper-resistant technique.

Existing Method :

$$\sum_{i=0}^{N} A_i + pN(N+1)$$

Our Method :

$$\sum_{i=0}^{N} A_i + (c+s)N$$

7. Conclusions

We discussed problems of the existing method, and we have proposed a new access control method of mobile agents to solve such problems. We have introduced a centralized management approach so that an authentication server will authenticate and authorize agents registered to the server, wherever they go. In our method, information required for authentication and authorization of an agent is securely managed and processed in the authentication server. Furthermore, for the purpose to enhance the security and to realize efficient access control, new interface modules called "secure stub" are proposed. In the evaluation result, program size of a secure stub is about 5 KByte and its throughput is 3.1 Mbps, therefore, it can be efficiently downloaded to the visited agent activating server and provide a communicating function between the visitor agent and the service without any processing burdens. We also evaluate capacity of an authentication server, and get the result that the capacity of the server is 50 transactions/sec, which is feasible to the practical network scale. Our proposal is significant to the advancement of the existing mobile agent systems supporting secure communications. As for our future research, we will continue to evaluate our idea in detail from the standpoints of system scalability where a real agent environment and agent services are simulated.

References

1) Chess, D.M.: Security issues in mobile code systems, Vigna, G. (Ed.), *Mobile Agents and Security*, pp.1–14, Springer (1998).
2) Sander, T. and Tschudin, C.F.: Protecting Mobile Agents Against Malicious Hosts, LNCS on Mobile Agents and Security (1998).
3) Hohl, F.: A Model of Attacks of Malicious Hosts Against Mobile Agent, *4th MOS'98 Secure Internet Mobile Computations* (1998).
4) Jansen, W.: A Privilege Management Scheme for Mobile Agent Systems, *1st International Workshop on Security of Mobile Multiagent Systems, Autonomous Agents Conference* (2001).
5) Wang, C., Hill, J., Knight, J. and Davidson, J.: Software Tamper Resistance: Obstructing Static Analysis of programs, Technical Report CS-2000-12, Dept. of Computer Science, Univ. of Virginia (2000).
6) Matsuoka, M., Akai, K., Matsumoto, T. and Takewaki, K.: A Case Study for Evaluating Tamper Resistance of Key-built-in Cryptographic Software by Engineers, Technical Report of IEICE, ISEC 2001-59 (2001).

(Received November 29, 2002)
(Accepted June 3, 2003)

Shinsaku Kiyomoto received the B.E. College of Engineering Sciences, and M.E. Institute of Materials Science, from Tsukuba University, Japan, in 1998 and 2000 respectively. He joined KDD (now KDDI) and has been engaged in the research on network security, cryptographic protocol, and mobile security. He is currently a research engineer of Network Security Lab. in KDDI R & D Laboratories Inc. He is a member of JPS, IEICE, and IPSJ.

Toshiaki Tanaka received the B.E. and M.E. degrees of Communication Engineering from Osaka University, Japan, in 1984 and 1986 respectively. He joined KDD (now KDDI) and has been engaged in the research on network security, cryptographic protocol, mobile security, digital rights management, and intrusion detection techniques. He is currently a senior manager of Network Security Lab. in KDDI R & D Laboratories Inc. He is a member of IEICE and IPSJ.

Koji Nakao received the B.E. degree of Mathematics from Waseda University, Japan, in 1979. He joined KDD (now KDDI) and has been engaged in the research on multimedia communications, communication protocol, secure communicating system and information security technology for the telecommunications network. He received IPSJ Research Award in 1992. He is a member of IEICE and IPSJ. He has been a part-time instructor in Waseda University and The University of Electro-Communications since 2002.