A Framework for User Privacy Protection Using Trusted Programs

Kenichi Takahashi
Institute of Systems & Information Technologies/KYUSHU
takahashi@isit.or.jp

Zhaoyu Liu
Dept. of Software and Information Systems, Univ. of North Carolina at Charlotte
zhliu@uncc.edu

Kouichi Sakurai, and Makoto Amamiya
Faculty of Information Science and Electrical Engineering, Kyushu University
sakurai@csce.kyushu-u.ac.jp, amamiya@al.is.kyushu-u.ac.jp

Abstract

The evolution of mobile technologies enables us to realize ubiquitous computing environments. In such environments, a user's mobile terminal manages the sensitive information and assists in various activities based on the user's information. At the same time, information leakage will become a serious social problem. In this paper, we propose a framework to protect the sensitive information of users in a manner they consider safe. In the framework, the user provides a trusted program that implements the manner he/she considers safe. The information recipient accesses user's sensitive information through this trusted program. In this manner, the user can protect his/her sensitive information. However, there exist several challenges in the realization of this framework. In this paper, we propose a method for generating a trusted program and discuss the solutions to other challenges.

1. Introduction

The Internet is increasingly becoming an essential tool for our business and daily life. Nowadays, we can access the Internet through not only personal computers but also cellular phones, PDAs, etc. The evolution of mobile technologies will enable all equipments to be connected to the Internet. This will enable us to realize ubiquitous computing environments [1]. In a ubiquitous computing environment, a user employs a mobile terminal that manages his/her sensitive information. The mobile terminal assists in his/her activities by communicating with service providers around him/her [2, 3].

On the other hand, various instances of information leakage have occurred, such as the cases of Yahoo! BB and CardSystems Solutions. These problems are caused by the carelessness and/or malicious intent of the people accessing the information, for example, notebook theft, and illegal possession of a USB device containing the information [4]. These problems can be solved by providing sufficient training to employees, using PCs that are unable to connect to external storage devices, etc. If a user wishes to protect his/her sensitive information, he/she needs to ensure that the service provider provides countermeasures against these problems. However, there is no guaranteed solution to achieve this, even if an information recipient takes severe countermeasures. Therefore, the user has to devise a way to protect his/her sensitive information by himself/herself.

Therefore, we propose a framework [5, 6] to protect sensitive information. The core concept of our framework is that a user should be responsible for the way in which his/her sensitive information is used. In other words, apart from the an information recipient, the information owner should also be able to decide the manner in which information is used and the protocol for transmitting the information, i.e., a user should be able to devise countermeasures for protecting his/her information.

In our framework, the user defines the protection policy. This policy specifies the way the information is used. A service provider, who is the information recipient, has a program referred to as the original program that handles user information. When the service provider requests sensitive information from a user, the user creates a program, referred to as the trusted program, from the original program according to his/her protection policy. The trusted program restricts the manner of information usage according to the protection policy. The user sends it to the service provider. The service provider accesses the information through the trusted program. Since the service utilizes the information through the trusted program that implements the manner specified in user's protection policy, the use of information by the service provider is restricted within the trusted program capability. Moreover, since the trusted program is created from the original program, it imitates the behavior of the original program. Consequently, the user prevents the service provider from using information for illegal purposes. Moreover, the responsibility is reduced, because the user also shares the responsibility. However, some challenges exist in the realization of our framework. In this paper, we propose a method for generating a trusted program and discuss solutions to other challenges.

The remainder of this paper is structured as follows. Section 2 describes related works. Section 3 introduces our framework. Section 4 proposes a method for obtaining a trusted program and provides examples of trusted program generation. Finally, section 5 discusses solutions to other challenges and section 6 concludes the paper.

2. Related works

Cryptographic algorithms such as symmetric and public-key algorithms and technologies based on these algorithms, such as digital signatures and public key infrastructure (PKI), have been proposed [7]. These algorithms and technologies aim at preventing message interception or identification of communication partners. Therefore, we can ensure message confidentiality, integrity, and availability against malicious exploitation by third parties; however, it is difficult to prevent the illegal use of the released information by the use of these algorithms. At the bottom of homepages, we often find a link concerning privacy, shown as "Privacy Policy" on the Yahoo! Web site, "Privacy" on that IBM Web page, etc. The page describes how the company treats users' personal information that it collects. However, the users cannot confirm whether the company actually abides by its privacy policy or not. Consequently, one has no choice but to believe that the company complies with its privacy policy.

The Platform for Privacy Preferences (P3P) [8] enables Web sites to express their privacy policies in a standard format that can be interpreted automatically by user agents. Thus, a user agent can automate decision making by comparing a company-defined privacy policy with user-specified privacy preferences. However, P3P does not provide technical assurance that the sites abide by their respective privacy policies.

The Enterprise Privacy Authorization Language (EPAL) [9] is a formal language to specify fine-grained enterprise privacy policies. It defines formal privacy authorization rules. Employees within an organization are required to comply with the EPAL policies. Thus, EPAL prevents the illegal use of information by the employees of the organization. However, EPAL does not specify to the users how the organization protects their personal information. Consequently, users cannot know whether the organization manages personal information securely or not.

Various researchers have attempted to provide right to information access or services based on trustworthiness [10, 11]. These researchers aim to develop trust relationships among users. However, it is difficult to define a general method for developing trust relationships; this is because trustworthiness depends on the users' conditions and/or situations. Moreover, a user simply trusts that an information recipient does not use his/her information for illegal purposes. Therefore, the information recipient can use the information in any manner, even for an illegal purpose.

Some researchers propose to protect user information by employing mediator support [12, 13]. Here, we have to provide a mediator(s) that can be trusted by both the communicating parties. However, the employment of a mediator appears difficult in real situations with countless users and service providers. Otherwise, a computational load becomes centralized at the mediator end.

Encapsulated Mobile Agent-based Privacy Protection (EMAPP) [14] protects the user's sensitive information by not showing it to outsiders. In EMAPP, each user is assigned an encapsulated space that manages his/her information. A service provider has a mobile agent that checks the user's sensitive information. When the user requests a service, the mobile agent migrates into the user's encapsulated space, checks his/her sensitive information, and sends only its result (which is permitted by the user) to the service provider. Since the sensitive information is not released outside the encapsulated space, the information is protected. However, the user may alter the result of a mobile agent. Consequently, the service provider cannot trust the result provided by the mobile agent.

3. Information protection framework

We propose an information protection framework to protect the user's sensitive information.

3.1. An overview

In our framework, service providers intend to employ user's sensitive information, therefore, they have a program, called original program, to handle user information. The original program consists of client and service programs, that are executed at the user and the service provider ends, respectively. The task of the client program is mainly to obtain user information and send it to the service provider. The service program receives the information from the client program and processes it. Based on the results of the execution of the client and the service programs, the service provider provides the requested service to the user. Thus, the information use and providing of service are realized by the communication between the client and the service program, i.e. communication between the user and the service provider (figure 1). Moreover, the service provider has a usage policy that defines what information the service provider needs, for what purpose, and how the service provider uses the

information for providing the service. The user has a protection policy that defines the manner in which the information is used, which he/she considers safe. The protection policy contains a program conversion rule; that converts an original program to a trusted program.

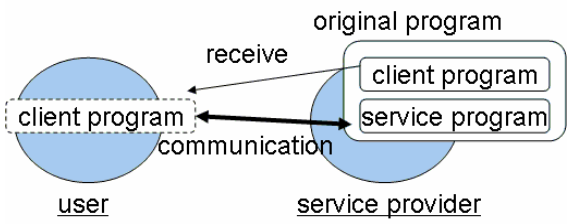


Figure 1. Service realization using the client and service program

When a user utilizes a service, he/she receives the usage policy and original program from the service provider. The user then creates a trusted program (which includes a trusted client program and a trusted service program) from the original program and the usage policy according to his/her protection policy (figure 2) and sends the trusted service program back to the service provider. The service provider employs the user information through the trusted service program. Here, the service is realized by the trusted program instead of the original program. Therefore, the information use by the service provider is restricted by the capability of the trusted program. Thus, the user prevents the service provider from an illegal use of his/her information. Moreover, the trusted program implements the way of using information in a manner the service provider prefers, because the trusted program is based on the original program.

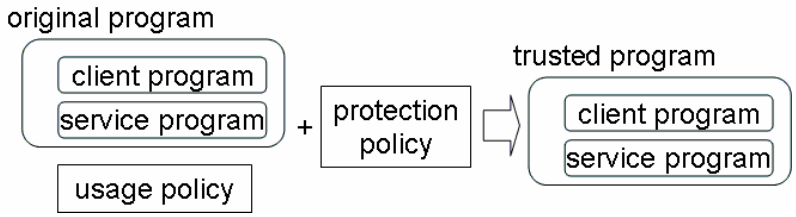


Figure 2. Trusted Program Generation

3.2. Challenges faced by the framework

We have provided an overview of the proposed framework to protect sensitive information. However, there are some challenges involved in its realization.

The first challenge is the generation of the trusted program: In the framework, the user creates a trusted program from an original program and the usage policy according to his/her protection policy. Therefore, we need to define the usage and protection policies for generating a trusted program from an original program.

The second challenge is to protect a trusted service program from a service provider: A trusted service program is executed at the service provider end. Therefore, the service provider can access and rewrite it and thus acquire sensitive information. Therefore, it is necessary to protect the trusted service program from illegal access and modification by the service provider.

The third challenge is to confirm the trusted program behavior: A service provider utilizes the user's sensitive information through a trusted program that is created by the user. However, the trusted program will be of no use to the service provider, if it fails to imitate the behaviour of the original program. Therefore, the service provider must verify certain aspects of the trusted program, such as its behavior, the trustworthiness of its creator (user), etc.

In this paper, we address the first challenge in the next section and discuss other challenges in section 5.

4. Trusted-program generation

First, in this section, we define the usage and protection policy and then provide an example of trusted program generation.

4.1. Usage and protection policy

The usage policy specifies what information the service provider needs, its purpose, and how the service provider uses this information for providing the requested service. Therefore, we define the usage policy as shown in figure 3. <info> specifies the information required by the service provider. <purpose> specifies the purpose of usage. <operation> specifies an operation performed on <info> in order to achieve <purpose>. <placement> has <op-place> and <va-place>. <op-place> specifies the line number of <operation> in the original program. <va-place> specifies the number of the line in which the value used in <operation> appears. Further, <va-place> provides an attribute of the value. "owned" represents the owner of the value, i.e., the user or service provider. "received from <line>" represents the value the user/service provider receives at <line> from the service provider/user, and "created from <names>" represents the value created from the values of <names>.

```
<usage policy> ::= <info><purpose><operation><placement>
<info>         ::= information necessary for providing the service
<purpose>      ::= P3P policy
<operation>    ::= "equal" | "greatherthan" | "lessthan" | ...
<placement>    ::= <op-place><va-places>
<op-place>     ::= <name><line>
<va-place>     ::= <name><attribute><line>
<name>         ::= the name of the instruction for <operation> or of the value
<line>         ::= a line number in the original program
<attribute>    ::= "owned" | "received from "<line> | "created from "<names>
```

Figure 3. Usage policy

The protection policy defines the manner in which the information must be used which the user considers safe. Therefore, we define the protection policy as shown in figure 4. <info> specifies the information to which this protection policy is applied. <preference> specifies the permitted purposes of <info> usage. <operation> specifies the permitted operation that can be applied to <info>. <conversion-rule> is a program conversion rule that converts an original program to a trusted program that implements the manner the user considers safe. The details of the program conversion rule are presented in the next section.

```
<protection policy> ::= <info><preference><operation><conversion-rule>
<info>              ::= information this policy applies
<preference>        ::= P3P preference
<operation>         ::= "equal" | "greatherthan" | "lessthan" | ...
<conversion-rule>   ::= described in Sect. 4.2
```

Figure 4. Protection policy

For example, only the equality operation is required by a service provider to confirm a password for user authentication. Therefore, the service provider defines a usage policy as follows: A password is required for user authentication, the password is used in an equality operation, and the line numbers where the password and equality operation appear in the original program. On the other hand, if a user wants to protect the password, he/she defines a protection policy as follows: Permit the password to be used only for authentication, only the equality operation is permitted, and the program conversion rule for the equality operation is defined.

4.2. Program conversion rule

A trusted program is created from an original program according to a program conversion rule. Since the trusted program aims to restrict the use of information by a service provider, the program conversion rule needs to convert some instruction in the original program to other instructions that comply with the user's protection policy. For this purpose, the program conversion rule consists of the following elements:

<exe-side>: specifies the entity (user or service provider) that initiates the trusted program.

<operation-rule>: defines the manner in which the result of <operation> is obtained. This rule is defined as follows:

<operation-rule> ::= <function>”->”<function>[<append-func>]

<append-func> ::= <operator><function>[<append-func>]

<operator> ::= “&&” | “||” | “xor” | ...

For example, if a rule is “f1(x) -> f2(y) && f3(z),” f1(x) is true if and only if both f2(y) and f3(z) are true.

<value-rule>: defines the values necessary for the implementation of <operation> and the method of its creation. This rule is defined as follows:

<value-rule> ::= <values>”from”<values>”by”<function>”at”<side>

<side> ::= "user" | "service provider"

For example, if a rule is “x from y, z by plus(y, z) at user,” then the user obtains value x from y and z by implementing the plus(y, z) method.

<value-attribute>: defines whether it is permitted to send the value to other entities or not. This rule is defined as follows:

<value-attribute> ::= <name><sendable>[<condition><to>]

<sendable> ::= true | false

<condition> and <to> are optional. The default value of <to> is user or service provider. If <sendable> is true and <condition> is satisfied, the value of <name> is sent to <to>.

<function-rule>: connects <function> defined in <operation-rule> and <value-rule> with the real function such as a Java function.

The user creates a trusted program by converting an original program according to the above-mentioned program conversion rule.

4.3. Example of the trusted program generation

We provide an example for creating a trusted program for user authentication. In this example, a user considers it safe to use the password in a hashed form for user authentication. Therefore, the program conversion rule will be defined as shown in figure 5. Of course, if the user wishes to protect the password by other ways, for example, the Caesar cipher encryption, the program conversion rule is defined accordingly.

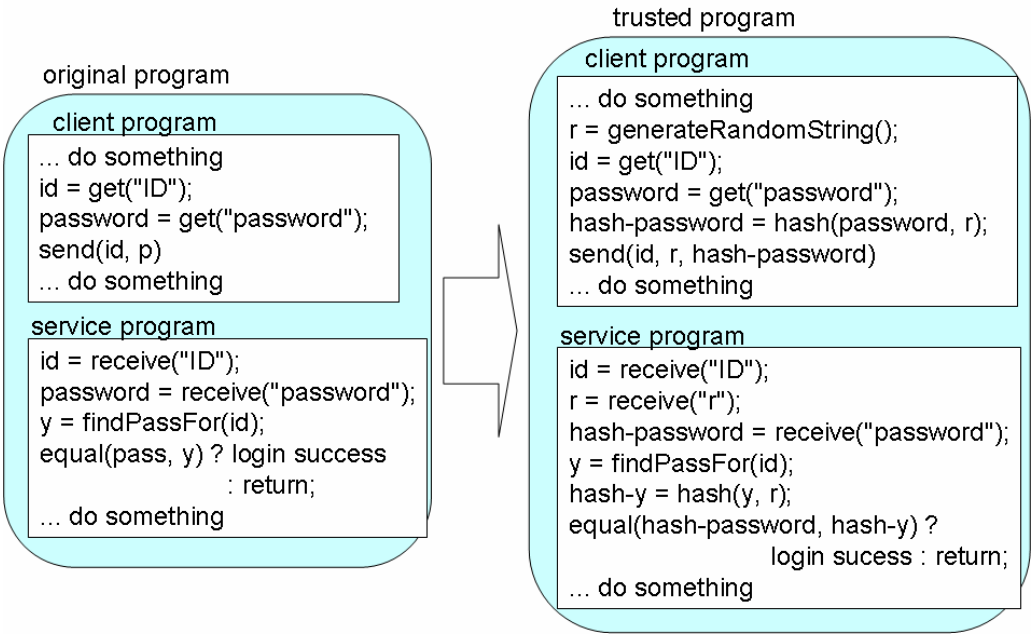
<exe-side> = user
<operation-rule> = equal(password, y) -> equal(hash-password, hash-y)
<operation-rule> = equal(y, password) -> equal(hash-y, hash-password)
<value-rule> = hash-password from password, r by $hash(password, r)$ at user
<value-rule> = r from NONE by $generateRandomString()$ at user
<value-rule> = hash-y from y, r by $hash(y, r)$ at service provider
<value-attribute> = password, false
<value-attribute> = hash-password, true
<value-attribute> = r, true
<value-attribute> = y, false
<value-attribute> = hash-y, false

Figure 5. Example of a program conversion rule for protecting passwords in hashed form

We assume the original program to be the program on the left side in figure 6. Then, according to the program conversion rule, the user converts the password and y in the original program to the hash-password and hash-y, respectively. Simply stated, equal(password, y) is converted to equal(hash-password, hash-y). Further, since <value-attribute> is true for the hash-password and r, they are sent from the client program to the service program, i.e. from the user to the service provider. As a result, the trusted program shown on the right side in figure 6 is created from the original program.

The service provider authenticates the user through the trusted program received from him/her. Since the trusted program uses the hash-password instead of the plaintext password, the service provider cannot use the password illegally. Thus, the user can safeguard the password against illegal use, even if the service provider is a Phishing site.

Figure 6. Example of a trusted program and its original Program



5. Discussion on other challenges

We defined the program conversion rule and provided an example of trusted program generation. Now, we consider the other challenges described in section 3.2.

5.1. Protection of a trusted program from a service provider

We introduced the information protection framework, which employs a trusted program for protecting the user's sensitive information. However, since the service provider executes the trusted service program, there is a possibility that the trusted service program can be altered and the sensitive information may be retrieved by the service provider from the trusted service program. Several methods for protecting the user's information (e.g. the example provided in section. 4.3) are tolerant to the abovementioned type of threats. However, we are afraid that this threat potentially exists in our framework. Therefore, we now discuss solutions to this problem. One solution is to use a tamper-proof device, and the other is to introduce trust relationships between the user and the service provider in the physical world.

5.1.1. Using a tamper-proof device: We assume that service providers possess a tamper-proof device that has a public key and a secret key for the public-key cryptosystem and a certificate for the public key provided by its manufacturer.

After a user creates a trusted program, he/she obtains the public key and its certificate from the service provider (the tamper-proof device). The user verifies the ownership of the public key by using a tamper-proof device made by a legitimate manufacturer. Subsequently, the user encrypts the trusted service program using the public key and sends it to the service provider. The tamper-proof device of the service provider decrypts the encrypted trusted service program by using the secret key. The tamper-proof device executes the trusted service program. Finally, the service provider can obtain the result from the tamper-proof device

(figure 7). Since the trusted service program is executed in the tamper-proof device, the service provider cannot view and alter the trusted service program. In this manner, we can protect the trusted program from the service provider.

However, this method requires that a tamper-proof device be provided to each service provider. In addition, there is the problem of resources for the tamper-proof device, e.g. the computational power, and memory spaces, etc.

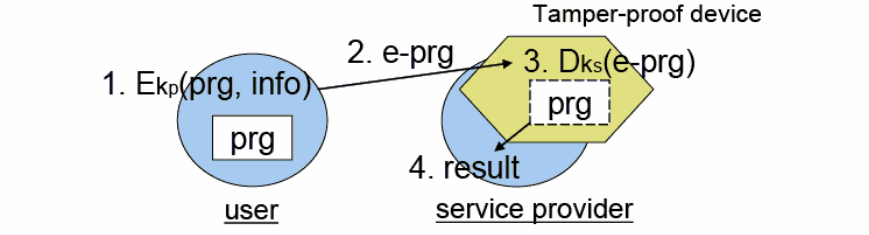


Figure 7. A method of using a tamper-proof device

5.1.2. Introduction of trust relationship in the physical world: Trustworthiness in simple words means that one believing the other. Hence, even if a user trusts a service provider, he/she cannot prevent the service provider from analyzing the trusted program. However, the risk can be reduced. Therefore, we consider introducing trust relationships in the physical world into the framework for reducing the risk of the trusted program being analyzed.

In the digital world, we cannot meet other people face to face. Therefore, it is easy for one person to play two or more entities. It means that even if one entity falls into disrepute, the reputation of other entities is not affected. Moreover, spoofing in the digital world is not very difficult. In fact, Phishing sites are becoming the source of serious social problems. This makes it difficult to build trust relationships in the digital world. On the other hand, we build trust relationships in the physical world by face-to-face communication. Therefore, we consider mapping trust relationships in the physical world onto the digital world. Although trust relationships in the physical world are unreliable, they appear more reliable than those in the digital world.

Suppose a user has a mobile terminal that assists in his activities. When the user meets person who provides a service on the Internet, he/she infers trustworthiness of the service provider by real communication. Subsequently, his/her mobile terminal receives an address (e.g. IP or URI) and a certificate of the service provider. The user's mobile terminal accesses the address and authenticates the service provider by using the certificate. Finally, the risk of the trusted program being analyzed is reduced depending on the trustworthiness in the physical world.

In addition, we can use our social knowledge. Even if a user cannot meet with a service provider face-to-face, he/she may evaluate the service provider according to his/her social knowledge. For example, the user cannot trust every online shop; however, he/she may trust

their portal site, such as eBay and Amazon. We can believe that the risk from the portal site will be smaller than the risk from each online shop.

Furthermore, the introduction of trust relationships in the physical world can solve the problem of resources such as the computational power, and memory space on a mobile terminal. For example, a user sets up a machine in his/her home. The user can then build complete trust relationship between his/her mobile terminal and his/her home machine. This is a kind of trust relationship in the physical world. It is apparent that the terminal and the home machine can easily share any information, such as the encryption key, certificate, etc. Therefore, the mobile terminal can commit complex computations to the user's home machine without special care. Hence, the mobile terminal is required to possess only capability to encrypt and transmit data to the home machine.

5.1.3. Other approaches: Another approach is to make the analysis of a trusted service program difficult. Mobile cryptography [15, 16] and software obfuscation [17, 18] are technologies applicable for this purpose. The former allows direct computations without decryption on encrypted functions; however, it is applicable only to polynomial functions and rational functions. Software obfuscation converts a program into another program that has a similar behavior, but it is more difficult to analyze. These techniques may be employed instead of using a tamper-proof device.

In addition, the release of a trusted service program in small parts may be allowed. A user can then release the subsequent part of the trusted service program, if and only if the service provider executes the current part precisely. Therefore, if the service provider does not execute the current part precisely, he/she cannot obtain the subsequent part of the trusted program. This makes the analysis of the trusted program difficult.

5.2. Confirmation of trusted program behaviour

When the trusted program behaves in a manner not expected by the service provider, the program has no value for the service provider. Therefore, the service provider has to verify whether the trusted program is created from the original program and a legitimate protection policy. We believe that this challenge can be solved by attaching a certificate issued by legitimate authorities to the program conversion rule. The proof carrying code (PCC) [19] is also useful for addressing this challenge.

Moreover, we have to consider the intellectual property of original programs. Service providers may not want to reveal their original program to others. Therefore, we may need to hide a part of the original program that is related to intellectual property.

6. Conclusions

We have proposed a framework for protecting user's sensitive information. The basic concept of the framework is to make use of information through a program called the trusted program that implements the information usage in a manner considered safe by the user. In the framework, a user offers a trusted program to a service provider. The service provider then makes use of user's sensitive information through the trusted program. In this manner, the user protects his/her sensitive information. In addition, because the user has the responsibility of protecting his/her sensitive information, the responsibility of the service providers is reduced.

However, the framework poses several challenges and we tackle them in this paper. We have defined the usage policy, protection policy and program conversion rules for trusted program generation. In addition, we have discussed the protection of the trusted program from the service provider.

7. Acknowledgments

This research was supported by Strategic International Cooperative Program, Japan Science and Technology Agency (JST), Strategic Information and Communications R&D Promotion Programme (under grant 052310008). The first author is supported by a Grant-in-Aid for Young Scientists (B), 18700076, and the second author is supported partially by NSF Grant 0406325.

8. References

[1] M. Weiser, "The Computer for the Twenty-first Century," Scientific American, pp. 94-104, 1991.

[2] H. Morikawa, T. Aoyama, "Realizing the Ubiquitous Network: The Internet and Beyond," Telecommunication Systems, Vol.25, no.3-4, pp. 449-468, 2004.

[3] T. Okoshi, et al, "Smart Space Laboratory Project: Toward the Next Generation Computing Environment," IWNA 2001, pp. 115-121, 2001.

[4] Japan Network Security Association, "Information Security Incident Survey Report ver.1.0," http://www.jnsa.org/result/index.html.

[5] K. Takahashi, K. Sakurai, M. Amamiya, "A Framework for Protecting Private Information through User-Trusted-Program and its Realizability," UISW 2005, LNCS 3823, pp. 433-442, 2005.

[6] K. Takahashi, K. Sakurai, "A Framework for the User-oriented Personal Information Protection," SAM 2006, pp. 12-18, Jun. 2006.

[7] D.R. Stinson (ed.), "Cryptography: Theory and Practice," Crc Pr I Llc, 1995.

[8] P3P project. http://www.w3.org/P3P.

[9] The EPAL 1.1. http://www.zurich.ibm.com/security/enterpriseprivacy/epal/.

[10] C. English, P. Nixon, S. Terzis, A. McGettrick, H. Lowe, "Dynamic Trust Models for Ubiquitous Computing Environments," UBICOMP 2002.

[11] G. Theodorakopoulos, J. Baras, "Trust Evaluation in Ad-Hoc Networks," WiSe04, pp. 1-10, 2004.

[12] Y. Karabulut, "Towards a Next-Generation Trust Management Infrastructure for Open Computing Systems," SPPC04, 2004.

[13] C. Pearce, P. Bertok, R. Schyndel, "Protecting Consumer Data in Composite Web Services," IFIP/SEC 2005.

[14] S. Yamada, E. Kamioka, "Access Control for Security and Privacy in Ubiquitous Computing Environments," IEICE Trans. on Comm., Vol.E88-B, No.3, pp. 846-856, 2005.

[15] T. Sander, C. Tschudin, "Protecting Mobile Agents Against Malicious Hosts,", Mobile Agents and Security, LNCS 1419, pp. 44-60, 1998.

[16] T. Sander, C. Tschudin, "On Software Protection via Function Hiding," Proc. of the 2nd International Workshop on Information Hiding, LNCS 1525, pp. 111-123, 1998.

[17] D. Aucsmith, G. Fraunke, "Tamper Resistant Software: An Implementation," Proc. of International Workshop on Information Hiding, LNCS 1174, pp. 317-333, 1996.

[18] T. Ogiso, Y. Sakabe, M. Soshi, A. Miyaji, "Software Obfuscation on a Theoretical Basis and Its Implementation," IEICE Trans. on Fundamentals, Vol. E86-A, No. 1, pp. 176-186, 2003.

[19] G. C. Necula, "Proof-Carrying Code,", POPL'97, pp. 106-119, 1997.

Authors

Kenichi Takahashi received the B.E., M.E. and D.E. degree in Computer Science and Electrical Engineering, Kyushu University in 1999, 2001 and 2004, respectively. He is a researcher of Institute of Systems & Information Technologies/KYUSHU. His research interests include security, ubiquitous computing and multi-agent system. He is a member of Information Processing Society of Japan, Society for Artificial Intelligence of Japan and IEICE.

Zhaoyu Liu received the Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign in 2001. He is an Assistant Professor of software and information systems at the University of North Carolina at Charlotte. His research interests include security, ubiquitous computing, and wireless networks. His recent research projects include energy-efficient sensor networks, security for ubiquitous environments, enterprise security, and mobile network security.

Kouichi Sakurai received the B.S. degree in mathematics from Faculty of Science, Kyusyu University and the M.S. degree in applied science from the Faculty of Engineering, Kyushu University in 1986 and 1988, respectively. He had been engaged in the research and development on cryptography and information security at Computer and Information Systems Laboratory at Mitsubishi Electric Corporation from 1988 to 1994. He received the Dr. degree in engineering from Faculty of Engineering, Kyushu University in 1993. Since 1994 he has been working for Department of Computer Science of Kyushu University as an associate professor, and now he is a full professor from 2002. His current research interests are in cryptography and information security. He is a member of the Information Processing Society of Japan, the Mathematical Society of Japan, ACM, IEEE and the International Association for Cryptologic Research.

Makoto Amamiya is a professor emeritus at Dept. of Intelligent Systems, Faculty of Information Science and Electrical Engineering, Kyushu University. He received his B.E. M.E. and D.E degree from Kyushu University in 1967, 1969 and 1978. After graduated, he worked at Electrical Communications Laboratories, NTT, until 1988. His research interests are programming languages and their processors, natural language understanding, data flow architecture, parallel and distributed processing, functional-type language, intelligent processing architecture and multi-agent systems. He is a member of IEEE CS, ACM, AAAI, IPSJ, IEICE, JSAI, JSSST. He is a fellow of IPSJ and IEICE.