Protection of Personal Information based on User Preference

Kenichi Takahashi[1], Takanori Matsuzaki[2], Tsunenori Mine[3],
Takao Kawamura[1] and Kazunori Sugahara[1]

[1]Graduate School of Engineering, Tottori University,
4-101 Koyama-Minami, Tottori 680-8552, Japan.
{takahashi, kawamura, sugahara}@ike.tottori-u.ac.jp
[2]Kinki University School of Humanity-Oriented Science and Engineering,
11-6 Kashiwanomori, Iizuka-shi, Fukuoka 820-8555, Japan.
takanori@fuk.kindai.ac.jp
[3]Faculty of Information Science and Electrical Engineering, Kyushu University,
744 Motooka, Nishi-ku, Fukuoka 819-0395, Japan.
mine@al.is.kyushu-u.ac.jp

ABSTRACT

Some of Internet services require users to provide their information such as their name, address, credit card number, and an ID-password pair. In these services, the manner in which the provided information is used is solely determined by the service providers. As a result, even when the manner appears vulnerable, users have no choice to refuse, but to allow such usage. However, the owner of the information is the user. Thus, users along with the service providers should be able to determine the manner.

In this paper, we propose such a framework that enables users to select. A policy defines the manner in which his/her information is to be protected, and its manner is incorporated into a program. By allowing a service provider to use the information through the program, the user can protect his/her information in his/her selected manner. Thus, in this scenario, both the service provider and the user can determine the type of protection for the user's information. Moreover, the responsibility of the service provider for information protection would be reduced as the user would determine the type of information protection. We show the validity of the proposed framework with an example scenario to protect user password.

KEYWORDS

Privacy, personal information, user preference, information security, framework

1 INTRODUCTION

Nowadays, we use various Internet services in our business and daily life. Some of these services require users to furnish personal sensitive information such as their name, address, credit card number, and an ID-password pair when they use these services. For example, an online shopping site may request user's address and credit card number for the shipping and the payment of an item. However, several cases of information leakage, such as those involving Yahoo! BB and CardSystems Solutions, have been reported. For example, a leaked credit card number may lead to the loss of money and a leaked name may lead to privacy issues. Thus, information leakage has now become a serious problem. Therefore, it is imperative that we exercise caution against providing sensitive information to Internet services.

We use a number of techniques such as encryptions, digital signatures and public key infrastructure (PKI) to secure sensitive information [1]. These techniques are effective in protecting sensitive information from malicious third parties. However, they are not effective against the misuse of the information provided by users to legitimate service providers. Once service providers obtain sensitive information from users, there is a possibility that this information can be misused (e.g., leak, abuse, or accident). The Platform for Private Preferences (P3P) [2] enables Web sites to express their privacy policies, which indicate how the company treats sensitive information collected from users, in a standard format that can be interpreted automatically by user agents. It, however, does not provide technical assurance that the sites will adhere to their respective privacy policies. Researches on trust and reputation [3,4] aim to developing trust relationships among users. They are not directly connected to the prevention of information abuse, further, difficult to define a general method for developing trust relationships.

In this paper, we propose a framework that enables users to select the manner in which their sensitive information is protected. In this framework, a policy, which defines the type of information protection, is offered as a Security as a Service [1]; a policy contains rules for replacing a part of a program in a manner so as to enable the protection of information; the user selects a policy that defines the manner in which his/her information is to be protected; the type of information protection defined by this policy is incorporated into a program. By only allowing a service provider to use the sensitive information provided by a user through this program, the user can protect his/her sensitive information in a manner selected by him/her.

Consequently, both the service provider and the user can determine the manner in which information is used so as to protect the user's information. Moreover, this reduces the responsibility of service providers as far as information protection is concerned, because user would also be involved in determining the manner in which his/her information is protected.

The remainder of this paper is structured as follows. Section 2 presents our framework and Section 3 shows an application scenario. Then, the related studies are shown in Section 4. Finally, Section 5 concludes the paper.

2 USER CUSTOMIZED DATA PROTECTION FRAMEWORK

Typically, a service provider provides a service only to those users, who possess specific information such as an ID-password pair, a name, and an address. Thus, a user is required to provide such information to the service provider. Subsequently, the user no longer has control over the information released to the service provider. To solve this problem, we propose a framework that enables users to customize the manner in which their information is protected; this

[1] SaaS usually denotes *Software* as a Service, but here, represents *Security* as a Service.

framework will be provided as a kind of SaaS.

2.1 Security as a Service

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [5]. It is a major concern for users that data and processes are entrusted to cloud service providers [6,7].

The SaaS is a type of cloud computing service. Such services are provided by McAfee [8], Panda Security [9], Yahoo! Japan [10], Rakuten Ichiba [11], etc. These companies provide various security services such as malware detection, URL filtering, and anonymous trading services, which work on the cloud instead of on client-installed software. Such services require users to entrust the management of their information to the cloud service providers. This is one of major concerns for users when they use cloud services. If we can confirm and verify that data and processes in the cloud are secure, then such the concerns could be diminished. However, this is very difficult to achieve because of reasons such as security, privacy, and company regulation. Thus, it is preferable a cloud service allows users to verify and confirm the security it provides.

In our framework, policies for the protection of users' sensitive information are provided as a SaaS. Policies are available in the form of text-based information and open to the public. Because anyone can view and verify the policies, we can consider that the frequently used policies are reliable. In addition, when a user wishes to protect his/her information in a particular manner, the framework enables the user to select a policy that defines such type of information protection. The manner defined in the policy selected by the user can be embedded into a program. The service provider can only make use of the user's information through this program. Consequently, our framework allows users to select the manner to protect their sensitive information by using the appropriate policies.

2.2 An Overview of Our Framework

A service provider has a program for offering services after verifying the information provided by users. We call this program the *original program*. To protect user information from the service provider in a manner selected by the user, the manner should be incorporated into the original program. In our framework, the part of the original program that processes user information is replaced with operations for the realization of the policy chosen by the user. We call this program the *customized program*. By only allowing a service provider to use the user's sensitive information through a customized program, the user can protect his/her information in a manner selected by him/her. The overview of our framework is illustrated in Figure 1.

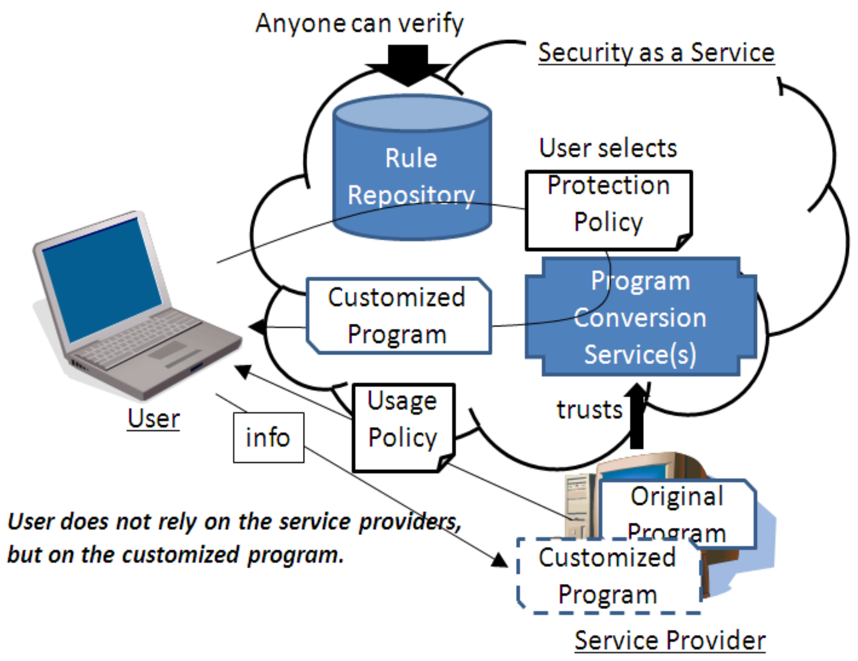


Figure 1. An Overview of Our Proposed Framework

The framework consists of users, service providers, a rule repository, and program conversion services. A user has sensitive information and requests a particular service to a service provider. The service provider has an original program and an *usage policy*. An original program provides the service to the users after verifying the user information. A usage policy defines what information the service provider requires, the purpose for which the information is used, and what operations are applied to the information. A rule repository and program conversion services are on the cloud as a SaaS. A rule repository stores *protection policies*, which define the type of information protection. Program conversion services provide a service for creating a customized program from an original program according to the protection policy.

When a user uses a service, he/she receives a usage policy from the service provider. The structure of the usage policy is shown in Figure 2.

The usage policy describes the information required by the service provider (`info`), the operations applied to the information (`operation`), variables used in the operation (`var`), and where the operation and variables appear in the original program (`location`). Thus, the user can know how his/her information will be used, where the information are operated in the original program.

Here, the operation shown in the usage policy is fixed solely by the service provider. If the user does not trust the reliability of the operation, he/she selects, from the rule repository, a protection policy that replaces the operation defined in the usage policy. Next, the user asks a program conversion service to create a customized program according to his/her selected protection policy. A program conversion service

```
info       ::= information the service provider requires
operation  ::= equal|greaterthan|…
var        ::= arg used in the operation
location   ::= (var|operation) [received|created] at line-x
```

Figure 2. Structure of Usage Policy

creates a customized program by replacing the programs indicated in `location` described in the protection policy. By allowing the service provider to use his/her information through the customized program, the user can protect his/her information in his/her chosen manner. Thus, the user, along with the service provider, can take up the responsibility of his/her information protection.

2.3 Rule Repository

A rule repository stores protection policies that define the manner in which information is protected. Anyone can obtain any protection policy from the rule repository, and thus, anyone can verify any protection policy.

Users who have the ability to verify a protection policy can make sure if the protection policy surely protects information. However, almost users have no ability to verify a policy. For such a user, a rule repository should provide some criteria that help users to judge if a protection policy is reliable or not. It would be possible to express the criteria as a score calculated from a number of uses, troubles and the policy verified. It is preferable to consider context information and characteristics of users for the score calculation. Then, protection policies with high score can be considered to be reliable. Further, if the administrator of a rule repository has the burden to verify a protection policy when the policy is registered in the repository, protection policies registered in would be considered to be reliable for users. Thus, almost users are not required to have no burden to verify a protection policy by themselves.

A protection policy is registered by volunteers. A volunteer could be a company or an individual. For example, a credit card company would register a protection policy for protecting a credit card number and recommends it to credit card users. Users can then use the recommended protection policy. In this case, users may not need to pay attention to whether the protection policy is reliable or not. Of course, if users are aware of a better policy to protect their information, they may/can choose that policy instead. In the case of using the recommended policy from the credit card company, the credit card company should take the responsibility of the abuse case of a credit card number, while the user chooses a policy by himself, he should take its responsibility.

2.4 Protection Policy

A protection policy consists of an objective operation, the explanation of the policy, and the program conversion rules to realize the protection. Program conversion rules consist of the rules to convert an operation and a value of the original program to other operations and values, respectively. Thus, the program conversion rules enable the creation of a customized program from the original program. The structure of a protection policy is shown in Figure 3.

```
operation       ::= equal | greaterthan | …
description     ::= how the protection policy protects information
op-change       ::= operation -> operation(s)
var-change      ::= var  created from operation(var(s))
                        [at user | service-provider]
```

Figure 3. Structure of Protection Policy

Rule `operation` specifies the targeted operation to which this protection policy can be applied. Rule `description` is written in natural language and describes the manner in which the protection policy protects user information. The protection manner written in `description` is defined using `op-change`, `var-change`, and `sendable`. Rule `op-change` is a rule for changing an operation to other operations to realize the protection written in `description`. For example, if `op-change` is "$f_a(x)$ `->` $f_b(y)$ `&&` $f_c(z)$", then $f_a(x)$ is replaced by the combination of $f_b(y)$ and $f_c(z)$. `Var-change` is a rule to define values used in the replaced operation by `op-change`. In above example, value `y` and `z` might be defined in. If `var-change` is "`y` *created from* `hash(x)` *at user*", value `y` is created from value `x` using a hash function on the user-side. `Sendable` defines whether it is permitted to send `var` to other entities (who is a user/service-provider when `var` is created at the service-provider/user side) under `condition`. A user understands a protection policy by viewing the text in `operation` and `description`, and asks a program conversion service to create a customized program according to the policy. Then, the program conversion service creates a customized program according to `op-change`, `var-change`, and `sendable`.

2.5 Program Conversion Service

A program conversion service provides a service for creating a customized program from an original program according to a protection policy. The program conversion service is basically available on the cloud, and a large number of such program conversion services exist. The program conversion service is selected by a service provider. Thus, a reliable program conversion service for the service provider would be selected. If there are no reliable program conversion services available to the service provider, the service provider can itself perform the task of program conversion service. Thus, the customized program created by the program conversion service would be relied on by the service provider.

A user, however, would not rely on the program conversion service, because it is selected by a service provider. Therefore, it is necessary for the user to verify the customized program is generated exactly as per a protection policy selected by him/her. Thus, we need 1) to show the service provider that the customized program was created by the program conversion service chosen by the service provider, and 2) to show the user that the customized program was exactly as per the protection policy selected by the user.

We simply solve the former issue 1) by using a digital signature to a customized program. For the latter issue 2), we put on a verification mechanism of a customized program. The user, of course, knows which policy is applied to the customized program. Therefore, the user verifies whether the protection policy chosen by the user is applied to the customized program or not. The verification mechanism of a customized program is shown in Figure 4, and is described as follows:

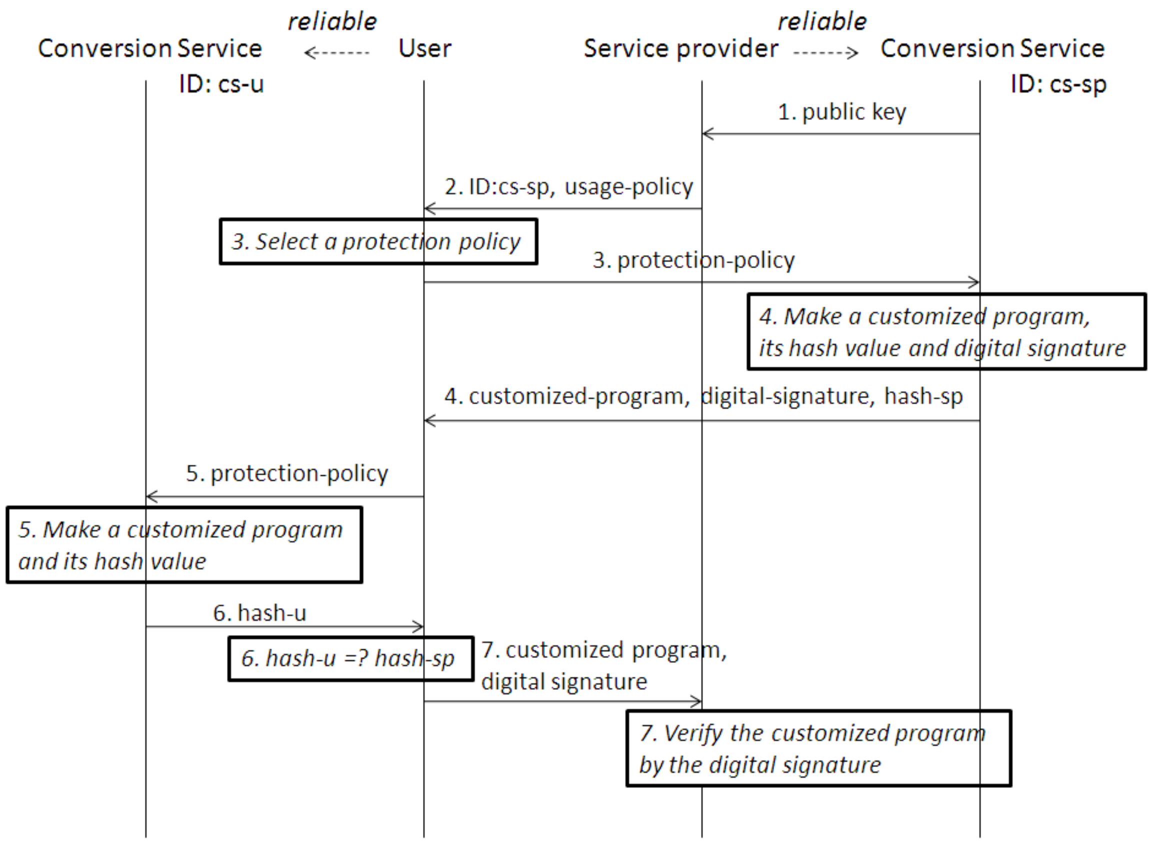

reliable
Conversion Service
ID: cs-u
User
reliable
Service provider
Conversion Service
ID: cs-sp
1. public key
2. ID:cs-sp, usage-policy
3. Select a protection policy
3. protection-policy
4. Make a customized program, its hash value and digital signature
4. customized-program, digital-signature, hash-sp
5. protection-policy
5. Make a customized program and its hash value
6. hash-u
6. hash-u =? hash-sp
7. customized program, digital signature
7. Verify the customized program by the digital signature

Figure 4.Verification Process of a Customized Program

1. A service provider selects a program conversion service and receives a public key (of a public key cryptosystem) from it.
2. The service provider informs the identifier of the program conversion service and the usage policy to the user.
3. Based on the usage policy, the user selects a protection policy from a rule repository. Then, the user asks the program conversion service to create a customized program according to the protection policy.
4. The program conversion service creates a customized program from the original program as per the

protection policy. After that, it generates a digital signature and a hash value for the customized program, and sends them back to the user.
5. The user selects a reliable program conversion service and asks it to create a customized program according to the protection policy. (In this step, the user is also permitted to perform the task of the program conversion service, if there are no reliable program conversion services available to the user.)
6. The user receives the hash value of the customized program from the program conversion service chosen by the user. If its hash value is the same with the hash value mentioned in step 4, the user can verify the customized program has been created as per the protection policy.
7. The user sends the customized program and its digital signature to the service provider. The service provider verifies the digital signature using the public key received in step 1. If the digital signature is verified to be correct, it is proved that the customized program has been created by the program conversion service chosen by the service provider.

Thus, both the service provider and the user can confirm that the customized program is created exactly as per the protection policy.

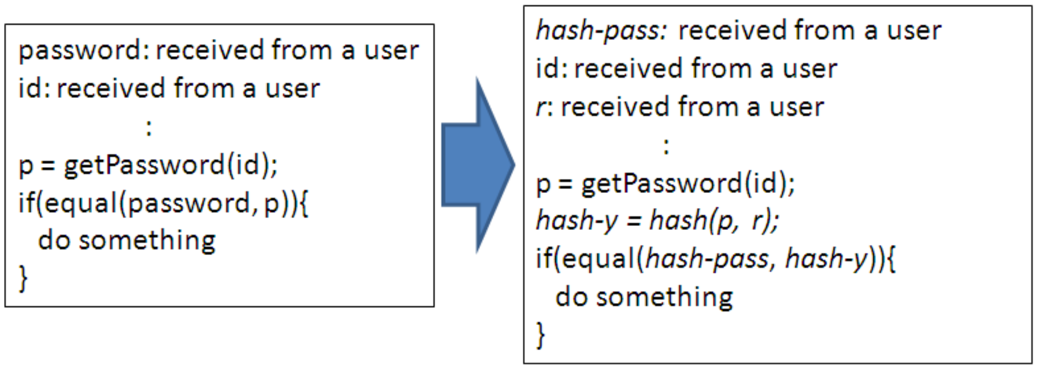
`operation`	= `equal(arg0. arg1)`
`description`	= *Can protect arg0 by using hash-value, when user has arg0, and service-provider has arg1.*
`op-change`	= *equal(arg0, arg1)* -> *equal(hash-pass, hash-y)*
`var-change`	= *r* created from *genRand()* at user
`var-change`	= *hash-pass* created from *hash(arg0, r)* at user
`var-change`	= *hash-y* created from *hash(arg1, r)*
`sendable`	= (*arg0, arg1, hash-y*), false
`sendable`	= (*hash-pass, r*), true

Figure 5. Protection Policy for Using Hashed Form Password

3 AN EXAMPLE SCENARIO

We describe a scenario that involves protecting a password from a service provider. In this scenario, a service provider requires a user's ID and password for user authentication. However, the user considers his/her password to be safe so that the password is only used in the hashed form, which is created according to a protection policy. At that time, the policy should be stored in a rule repository. Figure 5 shows an example of the protection policy for using hashed form password. Here, we should note that if the user believes another way to be safer, he/she will use that way for the protection of his/her password, even if it is vulnerable.
The service provider has an original program to check the user's ID and password. The original program would be as shown on the left of Figure 6. At

first, the user receives a usage policy of the password and the identifier of a program conversion service from the service provider. In this case, the usage policy would be
"`info=`*password*`,`
`operation=`*equal*`,`
`var=`*password*`,`
`location=`*password received at line 1*`,`
`location=`*equal at line 5*`.`"



```
password: received from a user
id: received from a user
            :
p = getPassword(id);
if(equal(password, p)){
  do something
}
```

```
hash-pass: received from a user
id: received from a user
r: received from a user
            :
p = getPassword(id);
hash-y = hash(p, r);
if(equal(hash-pass, hash-y)){
  do something
}
```

Figure 6. Left is an Original Program and Right is a Customized Program

The user selects a protection policy (shown in Figure 5) enabling his/her password to be used in the hashed form from a rule repository. Then, the user asks the program conversion service to create a customized program as per the protection policy.
The creation of a customized program is done as follows: *equal(password, p)* is replaced with *equal(hash-pass, hash-y)* according to `op-change` rule. Rule `var-change` describes the creation of a random number *r*, *hash-pass* from *arg0*, and *hash-y* from *arg1*. Here, the program conversion service can know *arg0* is *password* and *arg1* is *p* because *equal(arg0, arg1)* is *equal(password, p)*. In addition, because `sendable` of the *hash-pass* and *r* are true, *hash-pass* and *r* are sent from the user to the service provider. Finally, the customized program shown on the right of Figure 6 is created.
After that, as described in Section 2.5, the program conversion service sends the customized program with its hash value and digital signature back to the user. The user confirms that the customized program is created exactly as per the protection policy. The service provider verifies the customized program using the digital signature. Finally, the service provider executes the customized program. Thus, the user can protect his/her password in a manner that he/she believes to be reliable[2], even if the service provider is involved in phishing or other malicious activities.

4 RELATED STUDIES

Cryptographic algorithms such as symmetric and public-key algorithms as well as techniques based on them such as digital signatures and public key infrastructure (PKI) have been proposed [1]. These algorithms and techniques aim at preventing message interception or identification of communication

[2] Practically, *r* should be generated by the service provider in order to prevent a user from trying replay attacks.

partners. Thus, they ensure message confidentiality, integrity, and availability of the message against malicious exploitation by third parties. Such techniques, however, cannot prevent communication partners from abusing sensitive information released to them.

We often find a link on certain websites that points to the privacy policy adopted by that websites; this is termed as *Privacy Policy* on Yahoo!, *Privacy* on IBM, and so on. The privacy policy describes how the company treats sensitive information collected from users. The Platform for Private Preferences (P3P) [2] enables Web sites to express their privacy policies in a standard format that can be interpreted automatically by user agents. Thus, user agents can automate decision-making by comparing a company-defined privacy policy and user-specified privacy preferences. P3P, however, does not assure of that the sites will adhere to their respective privacy policy.

The Enterprise Privacy Authorization Language (EPAL) [12] provides fine-grained enterprise privacy policies, which employees within an organization are required to comply with. Whereas compliance to EPAL prevents the abuse of information by employees within an organization; it does not, however, provide an assurance to users that the methods used by the organization to manage their personal information are secure.

Various researchers have attempted to provide users with the right of information access or services based on trustworthiness [3,4]. Their researches were aimed to developing trust relationships among users. However, it is difficult to define a general method for developing trust relationships. This is because trustworthiness depends on user conditions and/or situations. Moreover, trust relationships are not directly connected to the prevention of information abuse.

Some researchers have proposed the use of mediator support to protect user information [13,14]. Mediators could be considered as a cloud computing service. It might be difficult to prepare such a mediator because the mediator should be trusted by both the sides involved in the communication.

Inada et al [15] and Miyamoto et al [16] propose methods to preserve privacy by forbidding the disclosure of the combination of information that may lead to the loss of anonymity. However, these methods cannot preserve privacy nor protect information when the loss of only a single piece of information (e.g., credit card number or password) causes a problem.

Encapsulated Mobile Agent-based Privacy Protection (EMAPP) [17] is a model in which a user's sensitive information is protected by not allowing service providers to access the information directly. 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 information. When a user requests a service, a mobile agent migrates into the user's encapsulated space, checks the user's information, and then sends only its result back to the service provider. Because sensitive information is not released outside the encapsulated space, the information is protected. However, it is matter to verify the security of the mobile agent because the mobile agent may be malicious. Moreover, a service provider may not be able to trust the result received from a

mobile agent, because a user could potentially alter the result sent back by it.
Further, these methods do not allow users to determine how their information is protected. However, our proposed framework enables users to select the manner in which their information is protected.

5 CONCLUSIONS

Some service providers request users to provide sensitive information. Once a user releases sensitive information to a service provider, the user has no control over it. The information would be under the control of the service provider. Thus, we must exercise caution against divulging sensitive information to service providers. To enable better protection of sensitive information, we have proposed a framework within which users can protect their sensitive information in a manner they believe to be reliable.
In our framework, rule repositories and program conversion services are offered as a *Security as a Service*. A user selects a protection policy that defines the type of information protection he/she desires from a rule repository. Then, the user asks a program conversion service to create a customized program that incorporates the desired information protection defined in the protection policy. Finally, by allowing the service provider to use his/her information through the customized program, the user can ensure that his/her information is protected in a manner chosen by him/her. Thus, the user, along with the service provider, can take up the responsibility of his/her information protection.
The future studies will include the evaluation of its efficacy. We also need to discuss in greater detail the structure of the usage policy and the protection policy.

ACKNOWLEDGEMENTS

This research has been supported by a Grant-in-Aid for Young Scientists (B), 23700098.

REFERENCES

1. D.R. Stinson, editor., Cryptography: Theory and Practice, Crc Pr I Llc, 1995.
2. P3P project, http://www.w3.org/P3P.
3. D. Xiu, Z. Liu, A Dynamic Trust Model for Pervasive Computing Environments, FTDC S 2004, pp. 80-85, 2004.
4. Y. Karabulut, Towards a Next-Generation T rust Management Infrastructure for Open Co mputing Systems, SPPC04, 2004.
5. P. Mell, T. Grance, The NIST Definition of Cloud Computing, http://csrc.nist.gov/groups/S NS/cloud-computing/
6. The result of questionnaire about Cloud Co mputing, http://jp.fujitsu.com/about/journal/voic e/enq/enq0905.shtml
7. 2010 Analysis Report of the Market of Clou d Service in Japan, http://www.idcjapan.co.jp/P ress/Current/20100603Apr.html
8. McAfee Security-as-a-Service, http://www.m cafee.com/us/saas/index.html
9. Panda Cloud Protection, http://cloudprotection .pandasecurity.com/
10. Yahoo! Auction - Safety Payment Service, h ttp://special.auctions.yahoo.co.jp/html/uketorigo/
11. Rakuten Safety Trading Services, http://event. rakuten.co.jp/anshin/
12. The EPAL 1.1, http://www.zurich.ibm.com/sec urity/enterpri seprivacy/epal/
13. G. Theodorakopoulos, J. Baras, Trust Evalu ation in Ad-Hoc Networks, WiSe04, pp. 1-1 0, 2004.
14. C. Pearce, P. Bertok, R. Schyndel, Protectin g Consumer Data in Composite Web Service s, IFIP/SEC 2005, 2005.
15. M. Imada, K. Takasugi, M. Ohta, K. Koyana gi, LooM: A Loosely Managed Privacy Prot ection Method for Ubiquitous Networking E

nvironments, IEICE Trans. on Comm., Vol.J 88-B, No.3, pp. 563-573, 2005.

16. T. Miyamoto, T. Takeuchi, T. Okuda, K. Ha rumoto, Y. Ariyoshi, S. Shimojo A Proposal for Profile Control Mechanism Considering Privacy and Quality of Personalization Servi ces, DEWS 2005, 6A-o1, 2005.
17. S. Yamada, E. Kamioka, Access Control for Security and Privacy in Ubiquitous Computi ng Environments, IEICE Trans. on Comm., Vol.E88-B, No.3, pp.846-856, 2005.