A Framework for Protecting Private Information Through User-Trusted-Program and Its Realizability

Ken'ichi Takahashi[1], Kouichi Sakurai[1,2], and Makoto Amamiya[2]

[1] Institute of Systems & Information Technologies/KYUSHU,
2-1-22 Momochihama, Sawara-ku, Fukuoka 814-0001, Japan
{takahashi, sakurai}@isit.or.jp

[2] Faculty of Information Science and Electrical Engineering, Kyushu University,
6-1 Kasuga-Koen, Kasuga-shi, Fukuoka 816-8580, Japan
amamiya@al.is.kyushu-u.ac.jp

Abstract. Thanks to the spread of mobile technologies, we will be able to realize the ubiquitous computing environment, in which equipment connected to the Internet assists users in their activities without special care on their part. Then, a function to protect private information is needed. This paper proposes a model for protecting private information. The basic idea of our model is to make use of private information through a program which a user trusts. A user offers a trusted program to a partner and compels a partner to make use of his private information through this program. In this way, a user prevents illegal use of his private information.

1 Introduction

Thanks to the spread of mobile technologies, people can access the Internet anytime through their cellular phones. In the near future, many equipments, for example cellular phones, refrigerators, microwave ovens, etc, will be connected to the Internet. This will enable the realization of the ubiquitous computing environment, in which equipment connected to the Internet assists users in their activities.

In the ubiquitous computing environment, there are a good many services which assist users. Then, it is difficult to find services which a user wants to use from among such a vast number of services. There are services which are universally available, and also are provided only to specific users. There are services which a user wants to use and also services which the user never use. Therefore, it is necessary to show services which the user can use and wants to use. Then, it is necessary to compare the substance of services with a user's private information. For example, the service in a liquor shop should be provided only for adults, not provided for children. Also, users who do not drink liquor will never use the service. Thus, the service should be offered only to users who are adults and drink liquor. Therefore, the user must reveal his age to the liquor shop for approval. Moreover it needs to judge whether he drinks liquor or not. But since this is private information, he may not want to reveal it. Also, some services may require users' private information. For example, online sales will require buyers' credit card information for payments. But credit card information is most important private information, and a user never wants it to be used for purposes other than the payment of his purchases. Therefore users must protect private information from illegal use by service providers.

As mentioned above, we need to provide private information to a service provider for service use and for the selection of services, and then we must protect private information. In this paper, we propose a model for protecting private information. The basic idea of our model is to make available private information through a program which a user trusts. A user offers his trusted program to a service provider and compels the service provider to make use of his private information through that program. In this way, a user will be able to prevent illegal uses of his private information by a service provider.

2 Related Work

Cryptographic algorithms, such as symmetric algorithms and public-key algorithms, and technologies based on them, such as digital signatures and Public Key Infrastructure (PKI), have been proposed. These algorithms and technologies aim at the prevention of message interception or the identification of communication partners. Therefore we can ensure message confidentiality, integrity and availability against malicious exploitation by third parties. But they are difficult to prevent illegal uses of released information by a communication partner.

At the bottom of the homepages, we often find links concerning privacy, shown as *Privacy Policy* at Yahoo!, *Privacy* at IBM and so on. These pages show how the company treats users' information that the company collects. Here, there are two problems. One is that users must read the privacy page carefully. Most people will not read the page, even when they provide their information. Another one is that we cannot confirm whether the company actually keeps the promises made on the privacy page or not. Consequently, we have no choice but to believe that the company will keep the promises written on the privacy page.

The Platform for Private Preferences (P3P) [4] 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 the comparison between a privacy policy and user-specified privacy preferences. So that, users do not need to read the privacy policies at every site they visit. P3P, however, does not provide technical assurance that sites act according to their privacy policies.

The Enterprise Privacy Authorization Language (EPAL) [7] is a formal language to specify fine-grained enterprise privacy policies. An EPAL policy defines format privacy authorization rules that allow or deny actions on data-categories by user-categories for certain purposes under certain conditions while mandating certain obligations. Employees within the organization are compelled to keep

EPAL policies. Thus EPAL prevents the illegal use of information by employees within the organization. But EPAL does not enable users to consent to how the organization protects their private information. Consequently, users cannot know whether the organization manages private information securely.

3 A Basic Model for Protecting Private Information

A service may need to check user's private information. Therefore, a user has to reveal some private information, such as a credit card number, his name and so on, if he wishes to use the service. Then, we must protect private information which we released.

3.1 Ways of Checking Private Information

The best way to protect private information is not to release private information to others (Fig. 1.a). Then private information will be checked by the user self. Therefore, a user does not need to worry about the risks of releasing private information. But we can apply this method only in situations where service providers do not need exact verification of users' private information. In other words, this method is not appropriate in situations that need a password check or a check of a right to use the service.

The next method is to release private information only to a trusted third party [3] (Fig. 1.b). The service provider commits the verification of private information to a trusted third party. And the user releases his private information to the trusted third party. Consequently, a service provider gets the result of the verification from the trusted third party without actually getting the private information. Then the user does not need to worry about the risks of releasing private information. But we must provide a third party that both of them can trust. It is difficult to provide a third party that any pair of users and service providers can trust. Even if we prepares some third parties which both of them can trust, we must worry that responsibility and the computational load are centralized in their third parties.

The last method is to release private information to a service provider (Fig. 1.c). If a user can trust a service provider, he may not need to worry about the risks of releasing private information. However, even if we assume a hierarchical PKI model, it is difficult to construct a mechanism that enable any user to trust any service provider. Therefore, we should also assume that a user cannot trust a service provider. If private information is released once to the other, the user cannot manage it. For this reason, it is risky to release private information to such an untrusted service provider, even if a user wants to use the service. Hence, we need a method that enables a service provider to verify private information (e.g. passwords and a right to use the service) and enables a user to prevent illegal use of private information.

To satisfy this dual requirement, we propose a method in which a user requires a service provider to make use of his private information through programs which

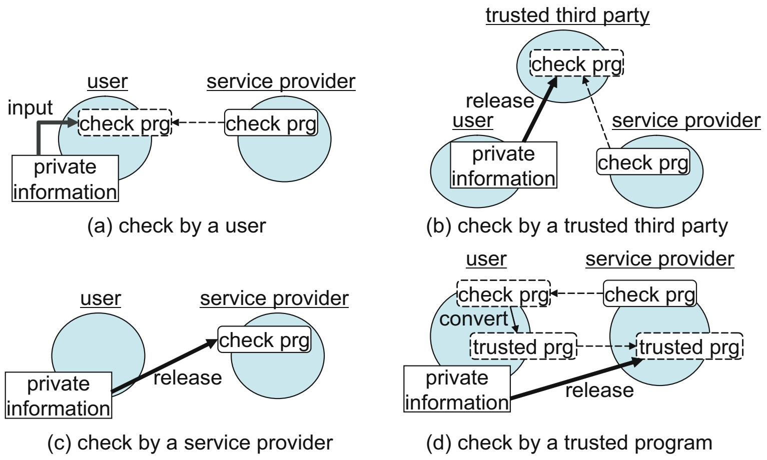

Fig. 1. Models for protecting Private Information

he can trust (Fig. 1.d). A user confirms that a program does not leak information, furthermore, is not used for purposes other than his wishes. Then, he requires the service provider to make use of his private information through the program. In this way, a user will be able to prevent illegal use of private information by the service provider.

3.2 The Public and Private Zone Model

We introduce the *public zone* and the *private zone* model [6] as a model for protecting private information based on verification by a user-trusted program. Our model is based on two agent systems, named KODAMA [8] and VPC [2]. In our model, users and service providers are represented as agents. An agent has a public zone and a private zone. The public zone is a freely accessible space and realizes flexible service use. The private zone manages and protects private information. And a *security barrier* exists between the public zone and the private zone. The security barrier has functions for restricting the access to private information and for the control of communications of a program which accesses private information. An overview of our model is illustrated in Fig. 2.

The Public Zone: In the ubiquitous computing environment, there are a good many services, which have a different method for its use. So, it is difficult to implement an agent which is able ab initio to use various services. Therefore, we define the *service program* and the *client program* as a pair.

The service provider creates a service program and a client program pair, and discloses a *public policy* in his public zone. A public policy consists of the client program and service attributes. Service attributes consist of a *description* for the explanation of the service, *access_info* which is information necessary for the service realization, *usage* for showing the purpose of *access_info* information use

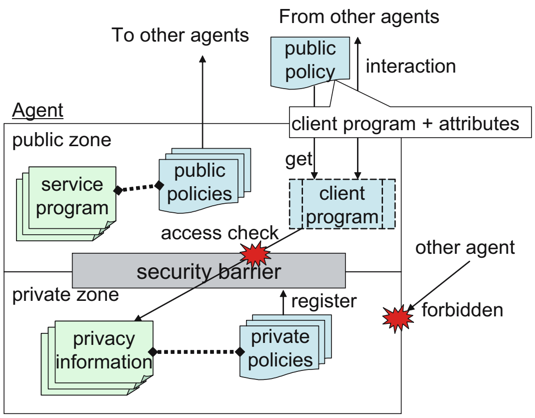

Fig. 2. The Overview of the Public and Private Zone Model

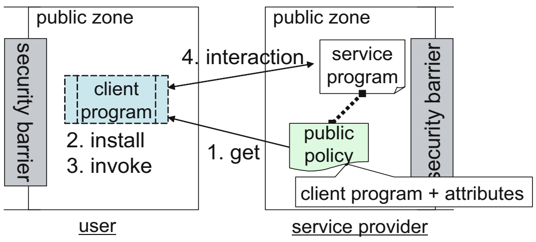

Fig. 3. The Flexible Service Use Mechanism

and *process* which is the utilization process of *access_info*. A user agent acquires a client program from a service provider agent and invokes it in his public zone. Then the service is actualized through communications, guided by the client program, between the service provider agent and the user agent (Fig. 3). In this way, a user agent can make use of various services without the implementation of explicit methods for the use of the various services.

The Private Zone: The private zone manages private information. An agent cannot directly access private information, but must access it through the public zone. Private information has a *privacy policy* which consists of *permission* for specifying access conditions, *allowable_partner* for specifying communications allowed to a program which accesses private information, *allowable_usage* for specifying permitted purposes of private information use, and *trusted_prg* for specifying trusted programs related with *allowable_usage*. Privacy policies are registered with the security barrier.

When a user wishes to use a service, the user agent acquires the public policy from the service provider agent and invokes its client program. Then, if the client program tries to access private information, the security barrier checks the access by *permission*. If the access is allowed, the security barrier returns

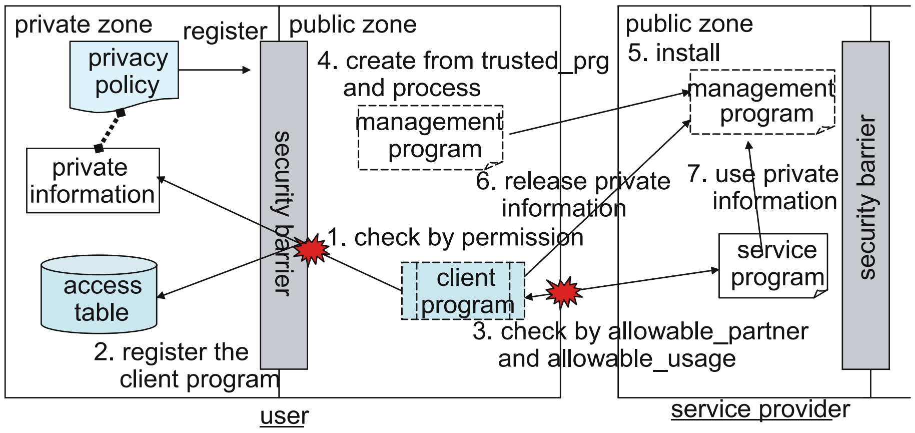

Fig. 4. The Protection of Private Information by the Security Barrier

its value and registers the client program in the access-table; if it is refused, an IllegalAccessException happens. After that, the security barrier monitors the communication of the client program registered in the access-table. When the client program communicates with other agents, the security barrier compares *usage* of the client program with *allowable_partner* and *allowable_usage* of the privacy policy. If it is refused, an IllegalCommunicationException happens; if it is allowed, the security barrier creates a *management program*, as a trusted program, from *process* of the public policy and *trusted_prg* of the privacy policy. And the user agent sends the management program to the service provider agent. After that, the user agent sends the private information (which is encrypted, not raw data) to the management program (Fig. 4). The service provider agent invokes the management program with this private information. In this manner, the user agent protects private information by restricting the access to private information and by requiring the use of the management program for the control of private information use.

3.3 A Simple Application Scenario

We show an example in which a user purchases a product. In the example, the user agent has private credit card information, and the service provider agent provides a product purchase service. The privacy policy of the credit card information is

```
permission=read_only, allowable_partner=publisher_only
  allowable_usage={payment:trusted_prg_only}
  trusted_prg={payment:trusted_payment_prg}.
```

Service attributes in the public policy are

```
description="Product Sales", access_info=creditcard
usage={creditcard:payment},  process={payment:payment_process}.
```

```
1: SalesProgram extends ClientProrgam{
2:    List product-list;  // the product list
3:    Agent owner;  // the publisher of this program
4:    public void main(){
5:       display(product-list);
6:       product = a product the user selected
7:       creditcard = accessToPrivateResource("creditcard");
8:       send(owner, {creditcard, price(product)}, process("payment"));
9:    }
10:}
```

Fig. 5. The Client Program of the Product Purchase Service

And the client program for the product purchase service is shown in Fig. 5.

Then a user purchases a product as follows.

1. A user agent acquires the public policy of a product purchase service from the service provider agent.
2. The user agent confirms whether the user can use it or not by the *description* and *access_info* of the public policy. Here, since the agent has credit card information to which access is allowed, the agent shows the product purchase service to the user.
3. When the user wishes to use the product purchase service (e.g. the user clicks "Product Sales" displayed on his terminal), the user agent invokes its client program. Then, the product list is shown to the user at line 5 in Fig. 5.
4. If the user selects a product, the client program tries to access the credit card information at line 7. Here, the access is allowed, since *permission* is read_only.
5. The client program tries to send the credit card information to the service provider agent for a payment of the product at line 8. To communicate with the publisher of the client program and to use the credit card information to pay for products are allowed by the *allowable_partner* and *allowable_usage.* But the credit card information is only allowed to be used by trusted programs. Therefore, the user agent creates a management program from the payment_process of *process* and the trusted_payment_prg of *trusted_prg* and sends it to the service provider agent.
6. The user agent sends the credit card information to the management program which was sent at step 5. Then, the service provider agent receives the payment by invoking the management program.

As shown above, the user agent confirms whether the service can be used or not at step 2, and restricts the access to the private information at step 4. Moreover, the user agent monitors the communication of the client program and allows the use of private information only by the trusted program (the management program). In this manner, user agents prevent the service provider agents from using private information for purposes which users do not desire.

4 Requirements for the Realization of Our Proposed Model

We have proposed a model for protecting private information, but we must overcome some challenges to realize our model. In this section, we focus on those challenges and discuss their resolvability.

4.1 A Method of Creating the Management Program

In our model, a user agent must creates a management program from *process* and *trusted_prg*. Here, an issure is how a management program can be created from *process* and *trusted_prg*. We suppose this can be realized by combining programs prepared in advance. For example, a user agent analyzes *process* of the public policy and extracts the part which uses private information. After that, the user agent replaces this part with a program specified in *trusted_prg*.

Consider a product purchase service. The product purchase service requires a credit-card payment and we assume its *process* is shown in Fig. 6. Here, a user agent may not be able to trust payment-prg at line 2. Therefore, the user agent replaces line 2 by trusted-payment-prg specified in *trusted_prg*. In this way, a user agent creates a management program from *process* and *trusted_prg*.

4.2 Protection of Management Programs

By compelling a service provider agent to use a management program, user agents protect private information. But if the service provider agent can rewrite the management program, the user agent cannot trust it any longer. Therefore, it is necessary to prevent the rewriting of management programs.

The easiest way to achieve this is to assume the use of anti-tampering devices. We implement public-key cryptography on anti-tampering devices, in which has a public and a secret key pair certified by a certificate authority. A service provider agent opens the public key to the public. A user agent encrypts the management program and private information with the public key and sends the encrypted them to the anti-tampering device of the service provider agent. The anti-tampering device decrypts the encrypted data using the private key and invokes their decrypted versions. In this manner, we can prevent the rewriting of a management program. But this requires that anti-tampering devices be installed in each service provider agent.

Another approach is to make the analysis of a management program difficult for a service provider agent. Mobile cryptography [5] and software obfuscation [1]

```
1 : purchase-program(creditcard, price){
2 :     payment-prg(creditcard, price); -------------------+
3 :     send a product;                                    | replace
4 : }                                                      |
                trusted-payment-prg(creditcard, price); <--+
```

Fig. 6. An Example of the Conversion of a Non-trusted Program into a Trusted One

are applicable technologies for this purpose. Mobile cryptography allows direct computations without decryption on encrypted functions. However, it is applicable only to polynomial functions and rational functions. Software obfuscation is a technique which converts a program into another program with a similar behaviour, but which is more difficult to analyze. However, its reliability and evaluation methods are not established.

The purpose of our model is to protect private information, not programs. In other words, a user agent must protect the management program only when private information is exposed to danger. For example, suppose that a management program deletes private information after the process has completed. Then private information can be protected, if it is possible to protect the management program until its process finishes. In this case, we will be able to use software obfuscation to protect the management program, even if its reliability and evaluation methods are not established.

Also, it may be allowed to interact with the user during management program execution and/or to release pieces of a management program little by little as the program progresses. Then it will be easier to protect the management program from service provider agents.

4.3 Confirmation of Management Programs

A service provider agent makes use of private information through a management program which is sent from the user agent. If the management program behaves in ways that are undesirable to the service provider agent, the program has no value for the service provider. Accordingly, a service provider agent has to be able to confirm the substance of the management program.

It is difficult to analyze a management program with no information. But a service provider agent knows the workflow of the management program by *process* and asks replaced parts from the user agent. Then, it may be possible to confirm whether the replaced parts work well or not. For example, a service provider agent may be able to confirm that the replaced part in Fig. 6 is warranted by the credit card company. Consequently, a service provider agent can rely on the management program.

Also, as a mentioned at Sect. 4.2, a management program is converted to a program (obfuscated program) which is difficult to analyze. So that, it is difficult to analyze the obfuscated program. Here, remember that our purpose is the protection of private information, not programs. Therefore, after finishing the process which requires private information, it is permissible to make the obfuscated program clear. Then the service provider agent will be able to confirm the de-obfuscated program. So, we suppose it is possible to ensure both the protection and confirmation of the management program.

4.4 Other Requirements

We must consider the possibility that a management program may be 'malware'. Therefore, we need to protect a service provider agent from malicious management programs. Also, to create private policies will be burdensome for users. Therefore, we have to consider who creates privacy policies. If information is supplied from other agents, these agents may be able to offer the privacy policy together with a private information.

5 Conclusions

This paper introduced a model for protecting private information and discussed its realizability. In our model, users and service providers are represented as agents who have a public zone and a private zone. The public zone is a freely accessible space for the realization of flexible service use. The private zone is a space for protecting private information by restricting access to private information and by the control of the communications of the client program which accesses private information. When a user needs to send private information to a service provider agent, the user agent compels the service provider agent to make use of private information through user's trusted program. In this manner, the user agent prevents the service provider agent from using private information for purposes which are undesirable to the user.

Acknowledgements. This research has been supported by the Telecommunications Advancement Foundation and Strategic Information and Communications R&D Promotion Programme under grant 052310008.

References

1. D. Aucsmith, and G. Fraunke. Tamper Resistant Software: An Implementation. In *Proc. of International Workshop on Information Hiding*, LNCS 1174, pp. 317–333, 1996.
2. T. Iwao, Y. Wada, M. Okada, and M. Amamiya. A Framework for the Exchange and Installation of Protocols in a Multi-Agent System. In *Proc. of Cooperative Information Agents 2001*, LNCS 2182, pp. 211–222, 2001.
3. C. Pearce, P. Bertok, and R.V. Schyndel. Protecting Consumer Data in Composite Web Services. In *Proc. of 20th IFIP International Information Security Conference*, pp. 19–34, 2005.
4. P3P project. http://www.w3.org/P3P.
5. T. Sander, and C. Tschudin. Protecting Mobile Agents Against Malicious Hosts. In *Mobile Agents and Security*, LNCS 1419, pp. 44–60, 1998.
6. K. Takahashi, S. Amamiya, and M. Amamiya. A Model for Flexible Service Use and Secure Resource Management. In *Advances in Grid Computing - EGC 2005*, LNCS 3470, pp. 1143–1153, 2005.
7. The EPAL 1.1. http://www.zurich.ibm.com/security/enterprise-privacy/epal/.
8. G. Zhong, S. Amamiya, K. Takahashi, T. Mine, and M. Amamiya. The Design and Implementation of KODAMA System. In *IEICE Transactions INF.& SYST.*, Vol. E85-D, No. 4, pp. 637–646, 2002.