Vol. 49 No. 2 IPSJ Journal Feb. 2008

Regular Paper

Design and Implementation of an Inter-Device Authentication Framework Guaranteeing Explicit Ownership

Manabu Hirano,[†1] Takeshi Okuda[†2] and Suguru Yamaguchi[†2]

Future networks everywhere will be connected to innumerable Internet-ready home appliances. A device accepting connections over a network must be able to verify the identity of a connecting device in order to prevent device spoofing and other malicious actions. In this paper, we propose a security mechanism for an inter-device communication. We state the importance of a distingushing and binding mechanism between a device's identity and its ownership information to realize practical inter-device authentication. In many conventional authentication systems, the relationship between the device's identity and the ownership information is not considered. Therefore, we propose a novel inter-device authentication framework guaranteeing this relationship. Our prototype implementation employs a smart card to maintain the device's identity, the ownership information and the access control rules securely. Our framework efficiently achieves secure inter-device authentication based on the device's identity, and authorization based on the ownership information related to the device. We also show how to apply our smart card system for inter-device authentication to the existing standard security protocols.

1. Introduction

Future ubiquitous networks will be connected to a large number of non-PC Internet-ready home appliances. A device accepting connections over a network must verify the identity of a connecting device digitally in order to prevent device-spoofing and other malicious actions. A digital identity for the device can be utilized by many kinds of useful access control systems. For example, a SIM (Subscriber Identity Module) is used in a mobile phone to maintain the subscriber's data.

From the point of view of model construction, It is necessary to distinguish between a device's identity and an owner's identity clearly because most devices have a single ownership or multiple ownerships. Therefore, we should consider a novel mechanism to guarantee the relationship between the device's identity and its ownerships. This guarantee is very important, because the actual access is caused not by the device's identity but by the owner's identity. The access control for the device should be processed based not on the device's identity but on the ownership information. This paper presents a new concept for an inter-device authentication and authorization framework.

Figure 1 shows the concept of the proposed inter-device authentication framework. Our proposal employs a tamper-resistant device, especially the smart card technology, to maintain securely the device's identity, the ownership information and the access control rule. Our novel smart card software can also execute device-specific security functions in a secure manner. A typical home appliance generally has only a minimum human interface, unlike a traditional PC with a keyboard and a display, making it difficult to input complex authentication and access control data. Our proposal for a novel smart card software with special configuration tools will enable manufacturers/users to set up the device's security configurations by separating the functions from the device itself.

This paper presents a new attempt to bind a device's identity and its ownership information efficiently. Thus, we propose a novel smart card software to realize inter-device authentication based on the device's identity, and authorization based on its binding. We also show how to apply our smart card to the existing standard security protocols.

The rest of this paper is organized as follows: Section 2 shows some practical device authentication mechanisms and their models. Section 3 describes some hardware-based security mechanisms with PKI support and explains why we employ a smart card in our proposal. Section 4 reviews the recent related work on device authentication. In Section 5, we introduce the inter-device authentication framework and its

†1 Toyota National College of Technology
†2 Nara Institute of Science and Technology

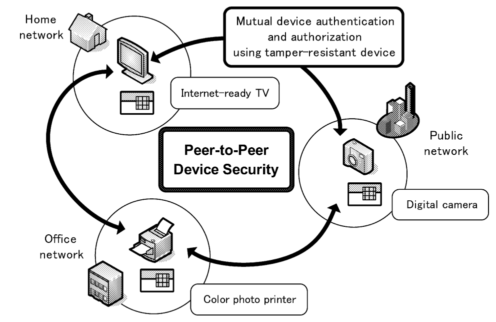

Home network
Mutual device authentication and authorization using tamper-resistant device
Internet-ready TV
Peer-to-Peer Device Security
Public network
Digital camera
Office network
Color photo printer

Fig. 1 Concept of an inter-device authentication framework.

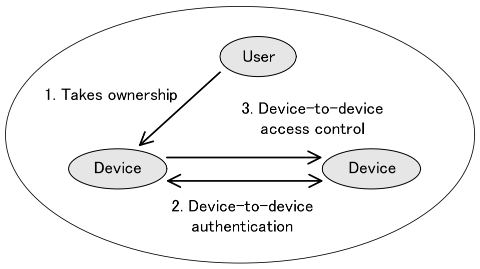

User
1. Takes ownership
3. Device-to-device access control
Device
Device
2. Device-to-device authentication

Fig. 2 Typical device authentication and access control model.

applications. In Section 6, we show the design of our framework. We present the prototype implementation of the novel smart card software and its configuration tools, and an authentication middleware system in Section 7. We show the result of performance measurements in Section 8. We discuss technical issues in Section 9. We conclude with a summary in Section 10.

2. Practical Model for Existing Device Authentication Mechanisms

We first review device authentication mechanisms in some major existing home network specifications. We then compare these specifications with a practical model for device authentication.

UPnP Security [1] UPnP (Universal Plug and Play) Security specification has a sophisticated mechanism to guarantee the personal ownership of devices. In the UPnP framework, the owner operates a special terminal device called "SecurityConsole" to set up security configurations on each device. The security-aware UPnP device has a public key pair and a password generated in the device itself. The device is identified by the SecurityID (hash value of the device's public key). The owner can assert ownership by inputting the password of the device from the SecurityConsole. The owner can also set up the ACL based on the SecurityID in each device by SecurityConsole via a network. After their respective setups, both devices can authenticate each other by a public key, and execute access control by the ACL and the SecurityID.

ECHONET [2] ECHONET (Energy Conservation and Homecare Network) is the standard specification for controlling home appliances proposed by major consumer electronics manufacturers in Japan. One advantage of the ECHONET is the range of the physical communication support such as the power line, the low-powered radio, and so on. ECHONET supports the ownership acquisition operation by inputting a serial key printed on the device. This serial key is generated and stored during the production phase. In ECHONET, the owner can set up a pre-shared key for each ECHONET device. Then, two devices can authenticate each other by that pre-shared key. ECHONET defines the original challenge and the response authentication protocol. ECHONET supports only a fixed 4-level authorization (User, Maker, Service Provider, Anonymous) with configurable access rules.

Bluetooth [3] Bluetooth is a short-range wireless communication system for electronic devices. Bluetooth has a secure simple pairing mechanism to take ownership of the device. Bluetooth has four association models depending on the I/O (display, keyboard and so on) capabilities of each device. To take ownership of the devices, the owner has to check and input the shared information (Bluetooth PIN, which is also called Passkey) displayed on a device. After the ownership acquisition operation, two devices generate a shared secret (called a link key) for future mutual authentication. However, Bluetooth does not support a device-to-device or device-to-user access control mechanism.

2.1 Comparison Details

In many existing authentication systems for the client-server paradigm, a user is simply authenticated by a server. However, in this paper, we discuss a security mechanism for an inter-device communication, for example, a communication between a home robot and a home server. We show the typical procedure of an inter-device authentication and an access control in **Fig. 2**. At first, an owner takes ownership of the device (establishing a secure relationship between the device and the

Table 1 Comparison of device authentication mechanisms.

	UPnP	ECHONET	Bluetooth
Taking ownership	Password	Serial key	Passkey (PIN)
Device-to-device authentication	Public key	Pre-shared key	link key
Device-to-device access control	SecurityID and ACL	Fixed level and ACL	-
Access control based on ownership	-	-	-

person). Next, the device-to-device authentication is executed. This step authenticates each *device's identity*. In the subsequent process, the device enforces access control rules based on *the ownership information* related to *the peer device's identity*. **Table 1** shows the comparison details of each mechanism. All the mechanisms described above support ownership acquisition and device-to-device authentication. Bluetooth does not support a user-configurable access control mechanism. UPnP and ECHONET support ACL-based access control mechanisms. Both mechanisms guarantee the relationship between the device and the owner in the ownership acquisition operation. Therefore, the device-to-device access control on both mechanisms also indirectly guarantees limited access control based on the ownership information. Both mechanisms work fully under conditions of single ownership of a device. However, since a device is often owned by multiple users, we should handle the device's identity and the ownership information separately.

2.2 Multiple Ownerships Model

Figure 3 shows an inter-device authentication/authorization model from the perspective of the relationship between a device and an owner identities. A device is mostly owned by someone. Therefore, the device's behavior is logically related, not with a *device's identity* but with an *owner's identity* (e.g., a human being or an organization). Thus, an access control mechanism for an inter-device communication has to have the ability to handle an *owner's identity*. The rest of this section compares the single ownership model with the multiple ownerships model shown in Fig. 3.

At first, (A) in Fig. 3 shows the single ownership model. Some existing authentication/authorization mechanisms described in Section 2.1 are a single ownership model. A device owner usually takes ownership on a device by inputting the password, the PIN, etc. Although this operation guarantees the *implicit ownership* of a device, this model does not handle an explicit *owner's identity*. In the single ownership model, an accessed device can only enable

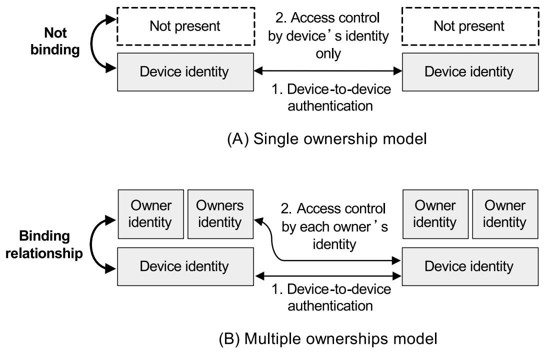

Not binding
Not present
2. Access control by device's identity only
Not present
Device identity
1. Device-to-device authentication
Device identity
(A) Single ownership model
Binding relationship
Owner identity
Owners identity
2. Access control by each owner's identity
Owner identity
Owner identity
Device identity
1. Device-to-device authentication
Device identity
(B) Multiple ownerships model

Fig. 3 Inter-device authentication/authorization model.

access control based on a peer *device's identity*. Next, (B) in Fig. 3 shows the multiple ownerships model we propose. This model supports multiple ownerships on a device, and it can distinguish each *owner's identity* explicitly. This model also supports a binding mechanism between a *device's identity* and *owner's identities*. In the multiple ownerships model, an accessed device can authorize a request based on a peer *owner's identity*.

In the single ownership model, a device can be owned by a single owner only. The device can only behave as a single device or a single owner in interaction with a peer device. If we adopt the multiple ownerships model, a device can be owned by multiple owners. The device can behave as multiple owners in interaction with a peer device. This means that an accessed device can authorize a request based not on a *device's identity* but on each *owner's identity*. For example, a TV set at home might be shared by a father, a mother and a child. The TV set can operate under each's ownership, and a peer device can authorize the TV set's request based on each'ss ownership. We show a more practical example of our proposal in Section 5.3. In this paper, we propose a novel framework that supports inter-device access control based on the explicit ownership. Our framework especially supports multiple ownerships on a single device, and it also guarantees the relationship between a device's identity and its multiple ownerships.

3. How to Store the Device and Owner Identities in a Secure Manner

The device and owner identities need to be protected from identity theft. Therefore, it is preferable to store highly confidential data such as private keys in a hardware-based secure storage. This kind of hardware-based system has tamper-resistant characteristics, and normally has an independent processor and a memory to execute security-purpose software securely, such as a smart card, a TPM (Trusted Platform Module), a HSM (Hardware Security Module), and so on. This section summarizes the major hardware-based security mechanisms.

PKCS#11 compliant Smart Card[4] RSA Laboratories have published the PKCS#11 cryptographic token interface standard. PKCS#11 specifies a platform-independent API set called *Cryptoki*. PKCS#11 provides an authentication mechanism not for a device but for a single user. PKCS#11 supports the RSA public key algorithm, the X.509 public key certificate, symmetric encryption algorithms such as AES, hash algorithms, and so on. *CryptoAPI* also provides the same kind of API as the one used in the Microsoft Windows environment.

Java Card compliant Smart Card[5] Sun Microsystems's Java Card platform technology enable developers to write original code for a smart card in Java programming language. The Java Card virtual machine in a Java Card-compliant smart card can execute a Java Card applet. Java Card supports fundamental cryptographic functions like the RSA public key algorithm, symmetric encryption algorithms, hash algorithms, and so on. Although the smart card has a limited memory and a slow speed of processor, developers can make an original applet that can execute on the smart card in a secure manner.

TPM[6] The TPM is defined as a microcontroller that stores keys, passwords and digital certificates. The TPM can store the user's or the device's identity like a private key. The TPM can also execute the user or the device authentication by TPM's digital signature functions. Moreover, a secure platform attestation mechanism guarantees the platform integrity by using Platform Configuration Registers (PCRs). In most cases, the TPM is on a PC's motherboard.

Table 2 shows a comparison of the hardware-based security mechanisms described above.

Table 2 Comparison of hardware-based security mechanisms.

	PKI auth.	Platform attestation	Programmable
PKCS#11 Smart Card	YES	NO	NO
Java Card Smart Card	YES	NO	YES
TPM	YES	YES	NO

The Java Card is a convenient choice to write the code and perform the prototyping for a secure protected execution environment. Thus we employ the Java Card platform technology to develop a novel smart card software which realizes our novel inter-device authentication framework, described in Section 2.2. Although the TPM's platform attestation is an essential component in every modern device's security, the platform attestation is outside the scope of this paper.

4. Related Work

Related research on device authentication/ authorization is available. Mayrhofer[7] shows a device authentication system specialized in context-based authentication using sensor data. His project provides an open source toolkit on Java with J2ME (Java 2 Micro Edition) to develop context-based device authentication software. Nguyen[8] proposes inter-device authentication using Identity-Based Cryptography (IBC). A typical authentication system based on a pre-shared key employs one common shared key for all the devices in the same domain. IBC-based authentication provides a method to share different keys for every two devices to improve security rather than one shared key for all devices. Fuji Xerox[9] announced the PKI-based device certificate service for their product, the multi-functional device *ApeosPort* in 2005. The device certificate profile proposed by Fuji Xerox has a product name, a serial number, and so on. They use their product as the starting point in the new challenge of device authentication/authorization services. Nicholson, et al.[10] describe a transient authentication for mobile devices. In transient authentication, a user has a wearable wireless token. A device detects the token and keeps an authenticated state while the user is close to the device. This mechanism guarantees the ownership acquisition operation by the wireless wearable token.

These studies deal with all the interesting aspects of device authentication. Our main contribution is the explicitly distinguishing and binding mechanism between the device's identity and the ownership information for an inter-device authentication, based on the PKI technology.

5. Overview of an Inter-Device Authentication Framework Guaranteeing Explicit Ownership

This section is a brief summary of a proposed inter-device authentication framework. In our proposal, a production-level device's identity is guaranteed by a X.509 Public Key Certificate[11),12)] generated by a manufacturer. The device's personal ownership is guaranteed by Attribute Certificates[13)]. An Attribute Certificate can isolate attribute data for access control from identity data in the Public Key Certificate. The Attribute Certificate is associated with the Public Key Certificate using a subject Distinguished Name (subject DN) field. In this paper, we employ Attribute Certificate as the key technology to distinguish and bind the device's identity and the ownership information securely. Hereafter, we abbreviate the X.509 Public Key Certificate as PKC, and the Attribute Certificate as AC.

Figure 4 shows the use case diagram for the proposal. During the production phase of a device, our framework enables manufacturers to register a production-level identity into a device's PKC. The production-level identity consists of the product's serial number and the manufacturer's information, which will never be changed. The manufacturer installs the PKC onto the device's secure protected area. After purchasing the product, our framework enables an owner to register the owner-level attribute (e.g., a distinguishable unique nickname assigned by the product owner) into a device's AC. An owner installs the AC and the Access Control List (ACL) of the device onto the device's protected area. The AC and the ACL can be changed by the device owner at any time using the personalization tool. Although a single device can register a single PKC (i.e., a single device's identity), it also can register multiple ACs (i.e., multiple ownerships). Consequently, an accessed device can authenticate peer devices by the production-level identity. Furthermore, an accessed device can also restrict a request from other devices based on the connecting device's owner-level attribute (ownership information) and its ACL. Although we should employ a fixed tamper-resistant chip embedded in a target device instead of a removable smart card, in our prototype implementation, all authentication and authorization information are stored onto the smart card. Therefore, for our prototype we guarantee the pairing between the smart card and the device by the smart card's PIN (owner's PIN) code.

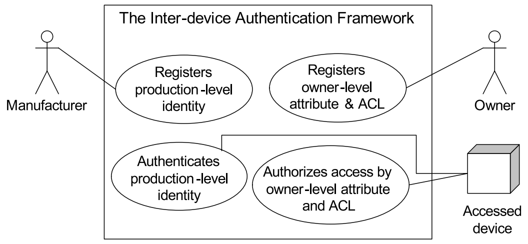

The Inter-device Authentication Framework
Manufacturer
Registers production-level identity
Registers owner-level attribute & ACL
Owner
Authenticates production-level identity
Authorizes access by owner-level attribute and ACL
Accessed device

Fig. 4 Use case diagram for the proposed system.

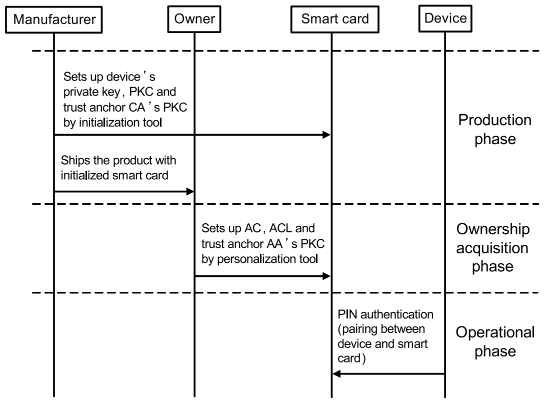

Manufacturer
Owner
Smart card
Device
Sets up device's private key, PKC and trust anchor CA's PKC by initialization tool
Ships the product with initialized smart card
Production phase
Sets up AC, ACL and trust anchor AA's PKC by personalization tool
Ownership acquisition phase
PIN authentication (pairing between device and smart card)
Operational phase

Fig. 5 Sequence diagram for the proposed system.

Figure 5 shows the sequence diagram for the proposed system. In the production phase, the manufacturer generates the device's RSA key pairs in a smart card, registers a production-level identity in the PKC, installs the PKC and the trust anchor CA's PKC onto the smart card. These operations are processed by our proposed initialization tool. The initialization tool also has a Certificate Authority (CA) function to issue a PKC from a public key. Our proposal assumes that the product is shipped with this initialized smart card. After purchasing the product, in the ownership acquisition phase, an owner registers his or her owner-level attribute (unique nickname for ownership information) in an AC, edits the ACL, and installs the AC, the ACL and the trust anchor AA's PKC on the smart card. These operations are processed by our proposed personalization tool. The personalization tool also has an Attribute Authority (AA) function to issue the AC from the PKC and an owner-level attribute. Finally, in the op-

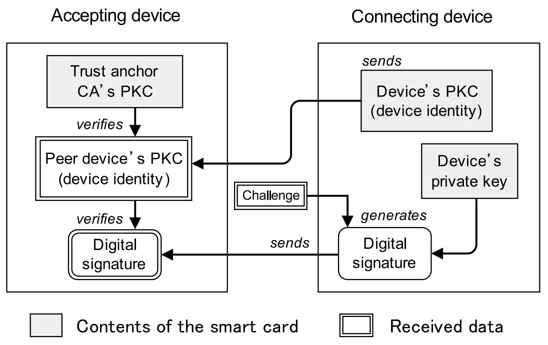


Fig. 6 Production-level identity authentication using PKC.

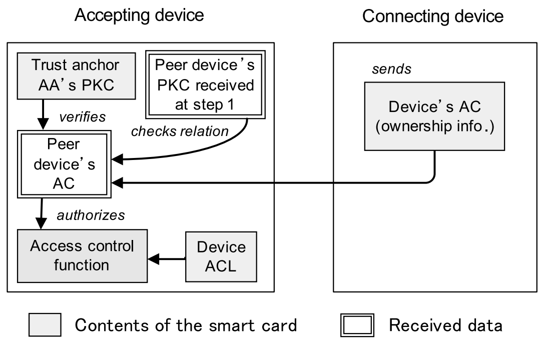


Fig. 7 Owner-level attribute authorization using AC.

erational phase, the owner attaches the smart card to the purchased appliance. Our proposal employs a PIN (an owner's PIN) code to ensure a coupling between the appliance and the smart card at this time. As a result, all appliances can utilize strong PKI authentication and authorization by means of both production-level identities and personal ownerships in each peer-to-peer connection. In the following sections, we describe the details of the operational phase.

5.1 Production-level Identity Authentication Using PKC

Figure 6 shows the inter-device authentication process. In an authentication protocol, the connecting device sends its PKC from the smart card to a peer accepting device. The received PKC is verified by the accepting device's trust anchor CA's PKC. If the verification is successful, then the accepting device sends a challenge to the peer connecting device. On the peer connecting device, the smart card generates a digital signature from the challenge and the private key, and returns it to the peer. The accepting device tries to verify the digital signature. If the verification is successful, then the authentication process is finished. For mutual authentication, the same process is required in the reverse direction. In Section 6.3, we show the design of the device authentication middleware system.

5.2 Owner-level Attribute Authorization Using AC

Figure 7 shows an authorization process based on the ownership information related to the device. This process must be performed after the authentication process. When the connecting device requests another device's operation, an application on the connecting device sends an AC from the smart card, which represents its ownership. An application on the accepting device verifies the received AC by the trust anchor AA's PKC, and checks the relation between the peer device's AC and the PKC received at the previous authentication step. Finally, the access control function in the smart card examines the authorization by the operation type, the AC's attribute and the ACL, and returns the authorization result to the application. For multiple ownerships on a single device, we propose that the AC (ownership) which is sent from the connecting device be automatically selected by a program based on a preconfigured rule. The rule maintains the mapping between each operation type and its ownership.

5.3 Example Usage

Figure 8 shows an example usage of our proposal. In this example, a security company operates a network-based security camera for an apartment building. The security camera is shared by the company and a resident of the apartment. The security camera is the *connecting device* in Fig. 7. The security company's video server and the resident's TV set are the *accepting devices* in Fig. 7. We consider the following two situations: First, the camera periodically sends pictures to the security company's video server to record pictures. Next, when the camera detects any unusual circumstances, the camera automatically interrupts the current TV program on the resident's TV screen and shows real-time camera pictures. Both actions are permitted by showing each ownership. Before they operate the system, the company and the resident each have to take ownership of the security camera by installing the AC. Each ownership (AC) is registered with the security camera, and it is bound to the camera's device identity (PKC).

Figure 9 shows ACL examples in the pro-

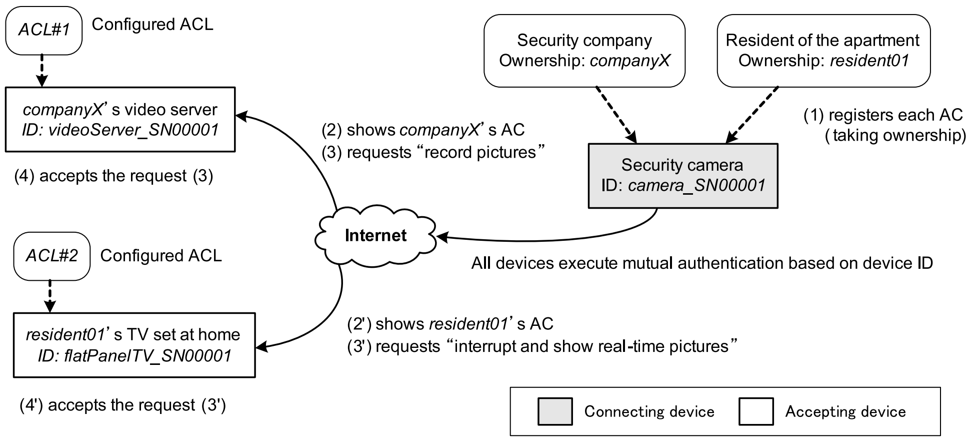


Fig. 8 Example usage.

```
Syntax:
  Deny/Allow <Operation> from <AC's attribute> on <PKC's subject DN>;
ACL#1:
  Deny from all;
  Allow "record pictures" from "companyX" on "camera_SN00001";
ACL#2:
  Deny from all;
  Allow "interrupt and show real-time pictures" from "resident01" on "camera_SN00001";
```

Fig. 9 Example usage of the ACL syntax.

posal. The ACL#1 shows ACL entries on the security company's video server. When the camera sends pictures to the company's server for recording, the camera and the company's server authenticate each other using the other device's identity (*camera_SN00001* and *videoServer_SN00001*), then the camera sends the AC for the *companyX*'s ownership related to the camera's identity (*camera_SN00001*). The company's server verifies the relationship between the PKC received at this authentication step and the AC, and examines the authorization of the camera's request based on the operation type (*record pictures*), the AC's attribute (*companyX*) and the ACL. The ACL#2 also shows ACL entries on the resident's TV set at home.

6. Design of an Inter-Device Authentication Framework Guaranteeing Explicit Ownership

Our framework described in the Section 5 consists of a novel smart card software for the devices' security, some configuration tools and an authentication middleware system for the smart card. In this section, we show the design of each component.

6.1 Novel Smart Card Software for Devices' Security

We employ a smart card with tamper-resistant characteristics to execute secure authentication and access control functions. **Figure 10** shows the design of our proposed novel smart card software. The smart card software consists of three modules. The authentication module provides a production-level device's identity authentication API using the device's private key and the PKC. The authorization module provides an owner-level access control API using the device's AC (ownership information) and the ACL. The proposed smart card software supports storage of multiple ACs because our framework supports multiple ownerships on a device. The security policy module is used to set up our authentication middleware system properly. The three modules are executed securely on the smart card. The device's private key is never leaked from the smart card because it is protected by a strong tamper-resistant storage. All API requests to the smart card are processed based on the ISO/IEC 7816-4 [14)] standard interface via a smart card reader.

Table 3 shows the authentication module API defined in the proposed smart card software. **Table 4** shows the authorization mod-

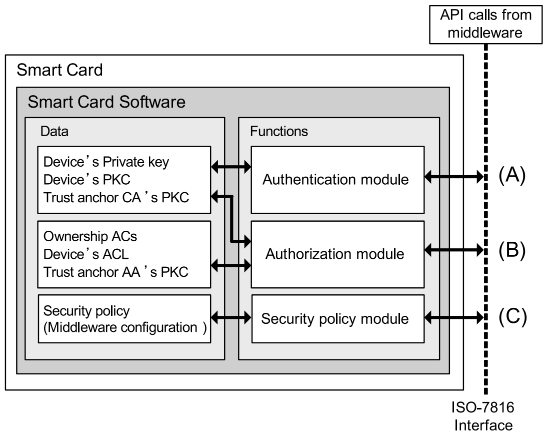


Fig. 10 Design of novel smart card software for device security.

Table 3 Authentication module API provided by our smart card software.

API	Description
geneateKeyPair	generate public key pair
retrievePublicKey	retrieve public key
storeDeviceCert	store device's PKC
retrieveDeviceCert	retrieve device's PKC
storeTrustAnchorCA Cert	store trust anchor CA's PKC
retrieveTrustAnchor CACert	retrieve trust anchor CA's PKC
getSignature	generate signature

ule API, and **Table 5** shows the security policy module API. We define two PIN types in our smart card software. The initialization tool executes *generateKeyPair*, *retrievePublicKey*, *storeDeviceCert*, *storeTrustAnchorCACert* in Table 3. These special APIs need an *administrator PIN* for the smart card. Other APIs can be executed with a smart card's *owner PIN*. Only the manufacturer knows the *administrator PIN*, and a device owner knows the *owner PIN*. The *owner PIN* is initialized by a manufacturer, and then the manufacturer discloses it to the purchaser. For example, to install an AC onto the smart card, the device owner has to input the *owner PIN* into the personalization tool. We assume that the *owner PIN* is shared by the device owners, therefore, the *owner PIN* can restrict unauthorized access to the smart card.

6.2 Configuration Tools for the Smart Card

We propose two configuration tools for the smart card described in Section 5.

Initialization Tool An initialization tool is used by the product manufacturer to register

Table 4 Authorization module API provided by our smart card software.

API	Description
storeOwnershipCert	store AC
retrieveOwnershipCert	retrieve AC
storeTrustAnchor AA Cert	store trust anchor AA's PKC
retrieveTrustAnchor AACert	retrieve trust anchor AA's PKC
storeDeviceACL	store device's ACL
getAuthorizationResult	get authorization result based on operation, peer AC's attribute, peer device's identity and ACL

Table 5 Security policy module API provided by our smart card software.

API	Description
storeSecurityPolicy	store security policy
retrieveSecurityPolicy	retrieve security policy
storeMWConfig	store middleware config
retrieveMWConfig	retrieve middleware config

production-level identity in the device's smart card. The Initialization tool allows the storage of the device's PKC and the trust anchor CA's PKC onto a smart card. The manufacturer inputs the product serial number as a Common Name (CN), and the manufacturer information as other attributes of the subject Distinguished Name (subject DN) in X.509 PKC. The DN consists of a CN, a country code (C), an organization (O), an organization unit (OU), and so on. The initialization tool must also have a CA functional capability to issue the X.509 PKC from the device's public key; communication functions based on ISO/IEC 7816-4 to read/write the public key and the PKC.

Personalization Tool A personalization tool enables the device owner to set up owner-level attributes (ownership information) onto the smart card. An owner can restrict the access from other devices by setting up the device's AC and ACL. The personalization tool must have AA functional capability to issue AC. The owner inputs a unique nickname as an attribute string in the AC to take his or her ownership of the device. A personalization tool reads the device's PKC from the smart card, and generates the AC related to the PKC's subject DN and his/her attribute. The owner also inputs the ACL to restrict the access based on the AC's attribute name. The personalization tool exchanges messages with the smart card on the ISO/IEC 7816 communication channel.

6.3 Inter-Device Authentication Middleware System

We propose an inter-device authentication middleware system using the original smart card software. In the current prototype system, our authentication middleware is executed on a Linux box (e.g., a micro server). We assume that the Linux box is attached to an information appliance by, for example, RS-232, USB or Ethernet, digital I/O. The Linux box with the smart card controls the information appliances. The pairing between the Linux box and the smart card is guaranteed by the smart card's PIN (owner PIN) authentication.

The two devices which are trying to connect to each other make it possible to execute a strong PKI authentication using the proposed middleware and the smart card. We employ the proven IKEv1 (Internet Key Exchange, version 1)[15] protocol to execute a mutual authentication between the two devices. The IKEv1 is used as an automatic key exchange to establish an IPsec[16] secure connection. We also employ the IPsec protocol to protect the communication channel between the two devices. Our proposed middleware (Customized IKEv1 program) invokes the authentication module API of the smart card. Consequently, the Linux box can authenticate the peer device's production-level identity described in Section 5.1.

Figure 11 shows the design of the proposed inter-device authentication middleware system. During the Linux box start-up, the IPsec policy management program calls the security policy module from the smart card. ((C) in Fig. 10 and Fig. 11) The smart card sends to the Linux box a security policy and some configurations for IPsec and IKE. The IPsec policy management program sets up the IP security policy onto the SPD (Security Policy Database)[17] in the Linux kernel. The preparation of the IP security policy is required to run the IPsec software and the IKE software.

When the device is trying to connect to another device, the IKEv1 program automatically begins to negotiate a new Security Association (SA) with another peer IKEv1 program. The SA consists of the required parameters such as a cryptographic session key, an expiration period of the session key, and an encryption algorithm to establish a secure communication channel. IKEv1 has two phases to establish an IPsec connection. The purpose of phase 1 is the mutual authentication and the establishment of the SA for IKE itself. The SA of this first phase is called the IKE-SA. In phase 2, the IKEv1 program negotiates with the SA for IPsec. The SA of this second phase is called IPsec-SA. We modify only the authentication process of phase 1.

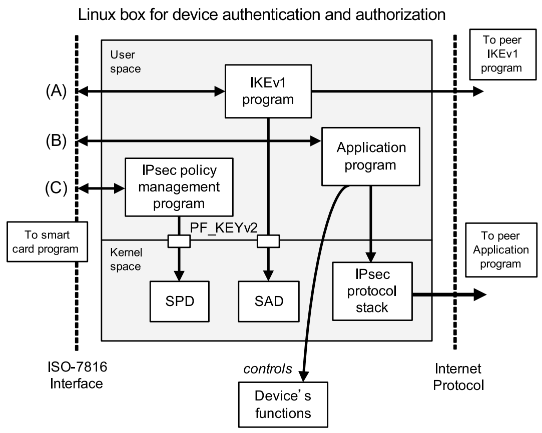


Fig. 11 Design of the inter-device authentication middleware system.

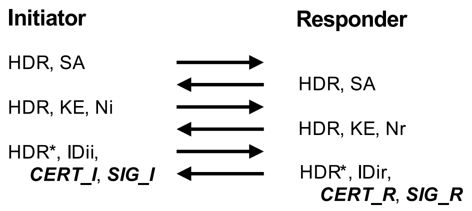


Fig. 12 Main mode with signature authentication on IKEv1 protocol.

During the negotiation in phase 1, the IKEv1 program calls the authentication module from the smart card ((A) in Fig. 10 and Fig. 11). We employ "main mode" as IKEv1's phase 1 exchange[15]. **Figure 12** shows the main mode with signature authentication on the IKEv1 protocol. In the main mode, the authentication module in the smart card generates a signature (SIG_I for the initiator, SIG_R for the responder) from the device's private key. Furthermore, the authentication module sends the device's PKC (CERT_I for the initiator, CERT_R for the responder) from the smart card to the IKEv1 program. Thus, the IKEv1 program can send them to the peer IKEv1 program. Consequently, the devices can authenticate each other based on the production-level identity in the smart card, and establish the IKE-SA. After establishing the IKE-SA, the IKEv1 program establishes the IPsec-SA by executing the conventional exchange in phase 2. The IKEv1 program stores the IPsec-SA parameters onto the SAD (Security Association Database)[17] in the Linux kernel. Finally, the communication channel between the two devices will be protected by IPsec.

7. Implementation

In this section, we show the prototype implementation of our proposal.

7.1 Novel Smart Card Software for Device Security

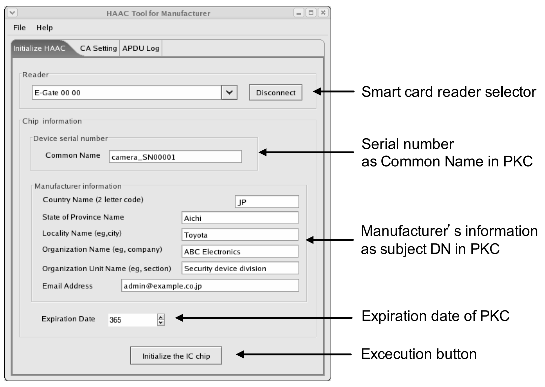
We have implemented the smart card software described in Section 6.1 running on an Axalto Cyberflex Access e-gate 32 k smart card and an Axalto e-gate token connector as a smart card reader. Our software employs the Java Card API 2.1.1. In this paper, we utilize the smart card as a small USB device, as shown in **Fig. 13**. We have implemented the following functions as a Java applet running on the smart card: an authentication module based on the device's identity, an authorization module based on the ownership information related to the device, and a security policy module to configure the middleware system.

7.2 Configuration Tools for the Smart Card

We have implemented the initialization tool and the personalization tool as a Java program running on a Linux operating system. **Figure 14** shows the initialization tool's GUI. Firstly, the manufacturer inserts a smart card with an initial state into a reader, and inputs an *administrator PIN*. The initialization program generates an RSA key pair in the smart card, and reads the public key from the smart card. The manufacturer can input the device serial number and the manufacturer's information using the GUI. The expiration date for the smart card can be enabled as the X.509 PKC's expiration date. Lastly, the manufacturer can store the device's X.509 PKC and the trust anchor CA's PKC onto the smart card.

Figure 15 shows the personalization tool's GUI. The personalization tool allows the storage of the device's AC, the ACL and a trust anchor AA's PKC. A user that purchases the product first inserts the smart card into a reader, and executes the personalization tool with an *owner PIN* on the PC. The user can then input his or her own attribute (nickname for ownership information), and can store the attribute as an AC. The user can also input ACL entries in text format by using a GUI. Finally, the AC, the ACL and the trust anchor AA's PKC can be stored onto the smart card. This step is the ownership acquisition operation for the device.

Fig. 13 Smart card as a USB device.


HAAC Tool for Manufacturer
File
Help
Initialize HAAC
CA Setting
APDU Log
Disconnect
Initialize the IC chip
Smart card reader selector
Serial number as Common Name in PKC
Manufacturer's information as subject DN in PKC
Expiration date of PKC
Excecution button

Reader: E-Gate 00 00

Chip information

Device serial number

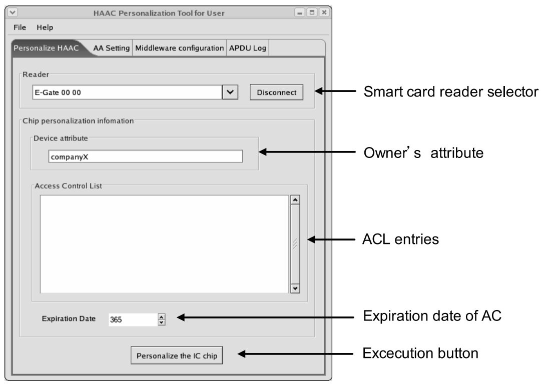
Common Name: camera_SN00001

Manufacturer information

Country Name (2 letter code)	JP
State of Province Name	Aichi
Locality Name (eg,city)	Toyota
Organization Name (eg, company)	ABC Electronics
Organization Unit Name (eg, section)	Security device division
Email Address	admin@example.co.jp

Expiration Date: 365

Fig. 14 Initialization tool for the smart card.



Reader: E-Gate 00 00

Chip personalization infomation

Device attribute: companyX

Access Control List

Expiration Date: 365

Fig. 15 Personalization tool for the smart card.

7.3 Inter-Device Authentication Middleware System Using IKEv1 and IPsec

Table 6 shows the software used with the middleware implementation described in Section 6.3. *USAGI*[18] is for the implementation of an IPv6 protocol stack with an IPsec function on the Linux platform. Our implementation employs the *USAGI*'s IPsec to secure the communication between the devices. *Racoon*[19]

Table 6 Software used with the middleware implementation.

Platform	Linux (kernel 2.6.11)
IPv6/IPsec stack	USAGI
IKEv1 program	Racoon (ipsec-tools 0.5.1)
IPsec policy management program	setkey (ipsec-tools 0.5.1)
PC/SC library	PCSC-Lite 1.2.9 beta 7

is the proven and major implementation of the IKEv1 protocol. We currently employ the IKEv1 protocol. To employ the new IKEv2 protocol [20], we have to modify an IKEv2 program to call the authentication module from the smart card. However, the modification of the IKEv2 program is simple, because it just requires replacing the digital signature functions with the smart card functions. Our implementation employs the 1,024-bit RSA-SHA1 signature/verification algorithm supported by the smart card. We utilize *setkey* [19] program as an IPsec policy management program to add, update, dump, or flush the SAD and the SPD entries in the Linux kernel. Our implementation extends the functions of the *setkey* program to load the IP security policy from the smart card. This process is only executed during the Linux box start-up. We employ the *PCSC-Lite* library [21] to utilize the smart card reader on the PC platform.

8. Performance Measurement

In this section, we show the results of the performance measurement of the authentication middleware system shown in Section 7.3. We prepared two Linux boxes with a configured smart card and deployed the middleware. **Table 7** shows the hardware specification of the Linux boxes. **Table 8** shows the respective processing times of the basic smart card functions used by the modified IKEv1 program. First, loading the device's X.509 PKC from the smart card corresponds to loading the device's PKC from the smart card by the connecting device in Fig. 6. Next, creating a signature for the payload in phase 1 of the IKE in the smart card corresponds to generating the digital signature by the device's private key in Fig. 6. In the measurement, we employ a 1024-bit RSA-SHA1 signature algorithm. **Table 9** shows a comparison between the proposed system and the conventional system in total time of mutual device authentication. This result expresses the total processing time to establish a connection in phase 1 of the IKE while using the main mode with signature authentication as shown in Fig. 12. The total processing time to establish the connection during phase 1 of the proposed IKEv1 program is 12.132 sec. The processing time includes the creation of a signature (SIG_I, SIG_R in Fig. 12) on the smart card (5.977 sec) twice for a mutual authentication. Our implementation supports the caching mechanism for the device's X.509 PKC (CERT_I, CERT_R in Fig. 12) at the IKE program initiation. Therefore, the result of the proposed IKEv1 program does not include the loading time of the device's PKC.

Table 7 Hardware specification of the Linux boxes.

CPU	Pentium M Processor 1.6 GHz
RAM	512 MB
Network	100Base-TX

Table 8 Processing time of the basic smart card functions.

	time [sec]
Load the device's X.509 PKC (942 byte) from the smart card	8.477
Create signature for IKE phase 1 payload in the smart card	5.977

Table 9 Comparison of the processing time of IKE phase 1 exchange.

	time [sec]
Conventional IKEv1 program	0.673
Proposed IKEv1 program	12.132

9. Discussion

We discuss three points: how to speed up the performance of the prototype implementation; the analysis and improvements on this study; and some PKI operational issues.

9.1 Performance Improvement

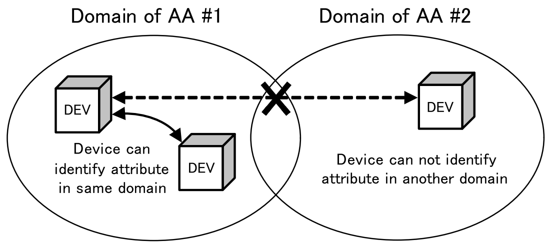
The current implementation of the proposed middleware is not fast (12.132 sec), because most smart cards have only a slow processor and a small memory to allow for secure tamper-resistant characteristics. Especially, the creation of a signature usually takes a long time. To improve the performance of the current implementation, we propose that the life time of the IKE-SA (i.e., the life time of the phase 1) extend as long as possible for periods not insecure, for example, about three days or one week. Our framework modifies only the authentication function in the phase 1 of the IKE negotiation; the subsequent phase 2 of the IKE is not modified and is fast enough. The authentication process of the IKE is only needed for an initial contact between the two devices. Moreover, typical appliances on a home network such as a TV and a PVR keep the same network topology for a long time. Therefore, extending the life time of the IKE-SA will help our implementation speed up the normal communication in each of the devices. This solution will provide enough security and performance for practical use. Furthermore, the processing speed of the smart card hardware may improve in the near future, and thus the processing time of our newly developed software will also decrease.

9.2 Analysis and Improvements

In many conventional authentication systems, the relationship between the device's identity and the owner's identity is not considered. Some authentication mechanisms described in Section 2 guarantee only the single ownership of a device. Our contribution in this paper is that our proposed framework can distinguish clearly between the device's identity and its ownership information by using the PKI technology. In our proposal, the ownership information (AC) is strongly associated with device's identity (PKC) based on PKI cryptographic techniques using trusted authorities of each certificate and their authorized digital signatures. Therefore, the authentication and authorization processes can be achieved efficiently. Thus, we have proposed a framework that enables the verification of the ownerships of a device. We have shown a prototype implementation for an inter-device authentication middleware system using devices' identities. We have also described an access control framework based on the ownership information related to the device.

In this paper, we employ a smart card to achieve original device security functions based on the PKI technology. To employ the PKI technology, our proposal is somewhat complex, so we have developed some configuration tools to reduce the user's burden in managing the smart card. An owner has to be able to operate the technology in spite of the complexity in the PKI. Further improvements are needed to enhance both the ease-of-use and the security.

From the perspective of the practical operation, the current initialization tool poses the problem that it requires substantial manual inputs from the manufacturer. We hope to address this issue in future work with a view to integrating automated registration procedures for high-volume manufacturing. In addition, the current personalization tool can execute on a local machine only. To make our proposal more convenient, we also have to consider supporting a remote update mechanism for a device's ACL on a smart card. In future work, we will develop the remote update mechanism on our personalization tool.


Domain of AA #1
Domain of AA #2
DEV
DEV
DEV
Device can identify attribute in same domain
Device can not identify attribute in another domain
Fig. 16 Multiple owner's AA.

9.3 PKI Operational Issues

Having proposed a PKI-based inter-device authentication framework, we will have to solve certain PKI operational issues like the following in our future work.

Multiple Manufacturer's CA Our proposal guarantees the device's production-level identity by using a PKC that is issued by the manufacturer's CA. If multiple manufacturers deploy our framework, then we must consider either of the following solutions: (1) Each manufacturer has a common root CA. The device stores the root CA's PKC in the smart card. (2) Each device stores all compatible CA's PKCs in the smart card. Solution (1) would require neutral organizations to guarantee the reliability of each manufacturer's CA. These are operational issues. For example, VeriSign already provides a device certificate service that supports a mechanism such as a chain of trust management [22]. We also have to consider an efficient mechanism (1) to update an expired PKC via a network without the physical smart card's recovery, and (2) to check the revocation status of the PKC, i.e., a mechanism like the CRL (Certificate Revocation List).

Multiple Owner's AA Our proposal guarantees the device's owner-level attributes (ownership information) by using the AC issued by the device owner's AA. If one owner's device tries to connect to another owner's device, then, without exchanging the AA's PKC, the device can not verify the second owner's AC and its attributes. If an owner brought the device to a different unknown domain of AA, this would cause a problem. **Figure 16** shows this problem. Possible solutions could be: (1) Each owner who will connect to another owner's devices has a root AA's PKC shared with friends, colleagues or members in another domain. (2) Each owner who will connect to another owner's devices stores all compatible AA's PKCs in the smart card. These are also operational issues, which we will consider in future work.

10. Conclusion

In this paper, we have analyzed the existing device authentication mechanisms. A single device has its device identity and is normally bound to a single owners' identity or to multiple owners' identities. We have therefore emphasized the importance of a distinguishing and binding mechanism between a device and its ownership information to realize practical inter-device authentication and authorization. In conventional systems, it is not possible to handle a device's identity and its ownership information separately, especially with multiple ownerships.

In this paper, we have shown an authentication and authorization framework for an inter-device communication which supports distinguishing between, and binding, a device's identity and its ownership information. Our proposal provides a production-level device's identity authentication using the PKC, and an access control mechanism based on the ownership information using the AC. The ownership information (AC) is strongly associated with device identity (PKC). This mechanism is based on PKI cryptographic techniques using trusted authorities of each certificate and their authorized digital signatures. Thus, the proposed framework enables the secure verification of the relationship between the ownership information and the device's identity. To realize our proposal, for the prototype implementation we have developed a novel smart card software to store securely highly confidential data such as a device's private key, and to execute a digital signature function and other functions. This paper has also presented the authentication middleware system using the proven IKEv1 protocol together with the proposed smart card. The configuration tools' purpose is to reduce the user's PKI operation.

In future work, we will apply our framework, especially the authorization mechanism in the proposal, to real home appliances. We also have to consider operational issues such as an expiration period for each PKC and AC, a cross-domain authentication and authorization, and a certificate revocation mechanism for our proposal. This is the first step towards the utilization of our framework. It is a fundamental mechanism for inter-device authentication and authorization which will contribute to the development of secure Internet-ready home appliances and their middleware systems.

Acknowledgments This work is partially supported by Exploratory Software Project, IPA, Japan.

References

1) Ellison, C.: UPnP Security Ceremonies Design Document For UPnP Device Architecture 1.0 (2003).
2) ECHONET Consortium: The ECHONET Specification, Version 3.2.1 (2005).
3) Bluetooth SIG: Specification of the Bluetooth System Version 2.0 + EDR (2004).
4) RSA Laboratories: PKCS11 Cryptographic Token Interface Standard, Version 2.20 (2004).
5) Sun Microsystems: Java Card 2.1.1 Specifications (2000).
6) Trusted Computing Group: Trusted Platform Modules Strengthen User and Platform Authenticity (2005).
7) Mayrhofer, R.: Towards an Open Source Toolkit for Ubiquitous Device Authentication, *Proc. PerSec 2007: 4th IEEE International Workshop on Pervasive Computing and Communication Security* (2007).
8) V. Nguyen, K.: Simplifying Peer-to-Peer Device Authentication Using Identity-Based Cryptography, *ICNS '06: Proc. International Conference on Networking and Services*, IEEE Computer Society, p.43 (2006).
9) Fuji Xerox: Introduction on PKI Technology and Device Certificate, Fuji Xerox Technical Report, No.15 (2005).
10) Nicholson, A.J., Corner, M.D. and Noble, B.D.: Mobile Device Security Using Transient Authentication, *IEEE Trans. Mobile Computing*, Vol.5, No.11, pp.1489–1502 (2006).
11) Housley, R., Polk, W., Ford, W. and Solo, D.: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, RFC3280 (2002).
12) ITU-T Recommendation X.509 (1997E): Information Technology — Open Systems Interconnection. The Directory: Authentication Framework (1997).
13) Farrell, S. and Housley, R.: An Internet Attribute Certificate Profile for Authorization, RFC3281 (2002).

14) International Organization for Standardization: ISO/IEC 7816-4 Identification cards — Integrated circuit cards — Part 4: Organization, security and commands for interchange (2005).
15) Harkins, D. and Carrel, D.: The Internet Key Exchange (IKE), RFC2409 (1998).
16) Kent, S. and Seo, K.: Security Architecture for the Internet Protocol, RFC4301 (2005).
17) McDonald, D., Metz, C. and Phan, B.: PF_KEY Key Management API, Version 2, RFC2367 (1998).
18) Kanda, M., Miyazawa, K. and Esaki, H.: USAGI IPv6 IPsec Development for Linux, *Proc. 2004 Symposium on Applications and the Internet-Workshops* (*SAINT 2004 Workshops*), IEEE Computer Society, p.159 (2004).
19) IPsec-Tools:
http://ipsec-tools.sourceforge.net/
20) C. Kaufman, Ed.: Internet Key Exchange (IKEv2) Protocol, RFC4306 (2005).
21) PCSC-Lite: http://pcsclite.alioth.debian.org/
22) VeriSign Device Certificate Service:
http://www.verisign.com/products-services/security-services/pki/device-certificate-service/index.html

(Received May 18, 2007)
(Accepted November 6, 2007)
(Online version of this article can be found in the IPSJ Digital Courier, Vol.4, pp.114–127.)

Manabu Hirano was born in Hokkaido, Japan. He received the M.E. degree in computer science from Nara Institute of Science and Technology (NAIST) in 2002. He worked for TOSHIBA Corporation since 2002 and worked on a mobile Internet-related business venture. Since 2004 he has worked for Toyota National College of Technology as a research associate. He is also now a Ph.D. candidate in NAIST. He also worked on the IPA's Exploratory Software Project in 2005 and 2007. His research interests include all aspects of ID management and its real-world applications.

Takeshi Okuda received the M.E. degree in information science from Osaka University in 1998. He is currently an assistant professor in the Graduate School of Information Science, Nara Institute of Science and Technology, Japan. His research interests include network security, mobile agent technology and multimedia application. He is a member of the IEEE.

Suguru Yamaguchi received the M.E. and D.E. degrees in computer science from Osaka University, Japan, in 1988 and 1991, respectively. From 1990 to 1992, he was an Assistant Professor in Education Center for Information Processing, Osaka University. From 1992 to 1993, he was with Information Technology Center, Nara Institute of Science and Technology (NAIST), Japan, as an Associate Professor. Since 1993, he has been with Graduate School of Information Science, NAIST, where he is now a Professor. Since April 2004, he has been an Advisor on Information Security, at Cabinet Secretariat, Government of Japan. His research interests include technologies for information sharing, multimedia communication over high speed communication channels, network security and network management for the Internet.