NAIST-IS-DD0661207

Doctoral Dissertation

Studies on Authentication and Authorization Mechanism for Inter-device Communication on Wide Area Network Environment

Manabu Hirano

Department of Information Processing
Graduate School of Information Science
Nara Institute of Science and Technology

February 7, 2008

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Manabu Hirano

Studies on Authentication and Authorization Mechanism for Inter-device Communication on Wide Area Network Environment[*]

Manabu Hirano

Abstract

This dissertation presents a novel security mechanism for an inter-device communication paradigm. Future ubiquitous networks will be connected to a large number of non-PC Internet-ready devices. This dissertation first introduces the new paradigm for inter-device communication and its characteristics. The inter-device communication paradigm has the following features: (1) mixed network architecture of both peer-to-peer and client-server, (2) single user multiple device paradigm, (3) distributed resources, (4) user-centric administration, and (5) autonomously controlled devices. To discuss security requirements for the inter-device communication paradigm, the dissertation shows some potential industrial applications like ITS, building automation and home networks. The security mechanism for the inter-device communication paradigm has to meet the following requirements: (1) authentication and authorization in a distributed environment, (2) separation between a device's ID and other attributes, (3) association between a device's ID and the attributes, (4) ID protection on distributed devices, and (5) independence from applications and implementations. Although many security mechanisms have been developed for the conventional client-server paradigm, it is difficult to directly apply such security mechanisms to a novel inter-device communication paradigm. An attempt at a new inter-device communication paradigm causes some security problems that need to be solved. This

[*] Doctoral Dissertation, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD0661207, February 7, 2008.

dissertation first presents an extension method of the standard network layer's security mechanism, in particular the IPsec protocol and the IKE protocol, for user-level applications. This attempt shows the feasibility study of the network layer's security mechanism on the inter-device communication paradigm. Next, this dissertation presents a novel inter-device authentication and authorization mechanism guaranteeing explicit ownership. The multiple ownerships model is the main concept of the proposal. The proposed model emphasizes the importance of the distinguishing and binding of the device's identity and ownership explicitly. The proposed mechanism employs PKI to guarantee the relation between the device's identity and the ownership by the cryptographic techniques of the PKI. Each device's identity and ownership can be expressed and verified based on the public key certificates and the attribute certificates. This dissertation presents novel security software running on smart cards to perform inter-device authentication and ownership-based authorization in a secure manner. The novel smart card software can manage the device's identity, ownerships and ACL securely. The proposed inter-device communication middleware is constructed on the standard network layer's security mechanisms, the IPsec protocol and the IKE protocol. To deploy the proposal in an actual environment, this dissertation also shows an initialization tool for manufacturers and a personalization tool for users. These tools can install and delete the device's identity, ownership and ACL on smart cards. This dissertation presents demonstration experiments for a TV device and a security camera to show the usability of the proposed inter-device authentication and authorization mechanism. The demonstration system consists of the proposed smart card and a micro server with the proposed middleware. The entire system works as a security proxy for the target appliance. This dissertation shows some discussions about life cycle management for devices, the hardware-dependence problem, group management methods for devices, prospective future applications and bridging architecture among heterogeneous systems. Finally, the dissertation summarizes the main contributions and future work.

Keywords:

Inter-device communication, Identity (ID) management, Ownership, Authentication and authorization, Access control, Public key infrastructure (PKI)

Contents

List of Figures

List of Tables

1. Introduction

In 1991, Mark Weiser first proposed the concept of “ubiquitous computing.” [1] He indicated that future sophisticated user interfaces will disappear from sight and be blended with a user’s surroundings. Ishi et al. showed the concept of “tangible bits” in 1997. [2] They showed a novel user interface using graspable media and ambient media. From the perspective of the user interface, their work essentially indicates that future networks will be connected to a large number of non-PC networked devices naturally. Future networks will consist of both such novel distributed devices and conventional client-server systems. Moreover, future networked devices will interact with each other and they will create novel network architecture based on inter-device communication. Federation of networked devices will allow federated services to be provided.

Until now, many security mechanisms have been developed for conventional client-server systems. A new paradigm of inter-device communication will cause some security problems that need to be solved. This dissertation demonstrates those problems and presents a novel security mechanism for inter-device communication systems.

1.1 New Paradigm for Inter-device Communication

This section first introduces a new paradigm for inter-device communication. Figure 1 shows the conventional client-server paradigm. A user utilizes the client computer to connect to the server computer. In the client-server paradigm, the administration point and the security mechanism are located on the server computer. From that viewpoint, the client-server paradigm makes a centralized administrative network. MIT’s Compatible Time-Sharing System [3] is an historically important client-server system from the 1960s. Many users used centralized computing resources using this time-sharing system. Even now, most information systems are constructed based on the client-server paradigm.

Figure 2 shows a new paradigm for inter-device communication. A user has many networked devices. Thus, this new paradigm reverses the number of computers and the number of users in conventional client-server paradigm. This dissertation assumes that each networked device is connected to the others and

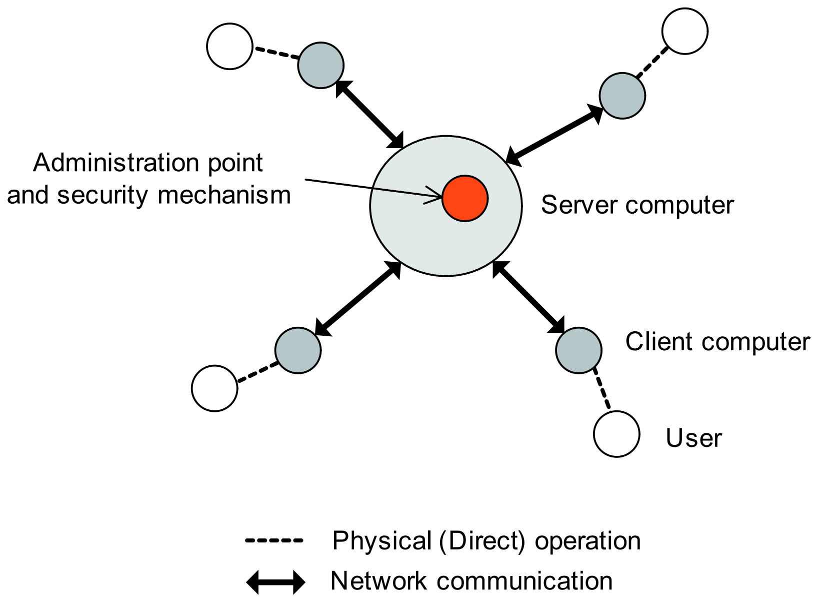

Administration point and security mechanism
Server computer
Client computer
User
Physical (Direct) operation
Network communication

Figure 1. Client-server paradigm

the federated devices provide some useful services. Although the security mechanism on the client-server paradigm is centralized, the security mechanisms for inter-device communication have to be distributed. Each user has to manage the administration point for her or his devices. Each networked device has its own resources to be protected.

1.2 What is Inter-device Communication?

This section briefly defines the basic architecture of the inter-device communication system. Conventional network architecture is known as *client-server*. Most applications on the Internet employ the *client-server* architecture. For example, WWW (World Wide Web) and e-mail are typical applications employing the *client-server* architecture. This dissertation assumes that future networked devices will interact with each other and that the networked devices can be federated for a variety of purposes. Therefore, the networked device has both the *client* and the *server* functions. From the perspective of network architecture, inter-device communication is represented as the *peer-to-peer* model. Figure 4 shows the basic

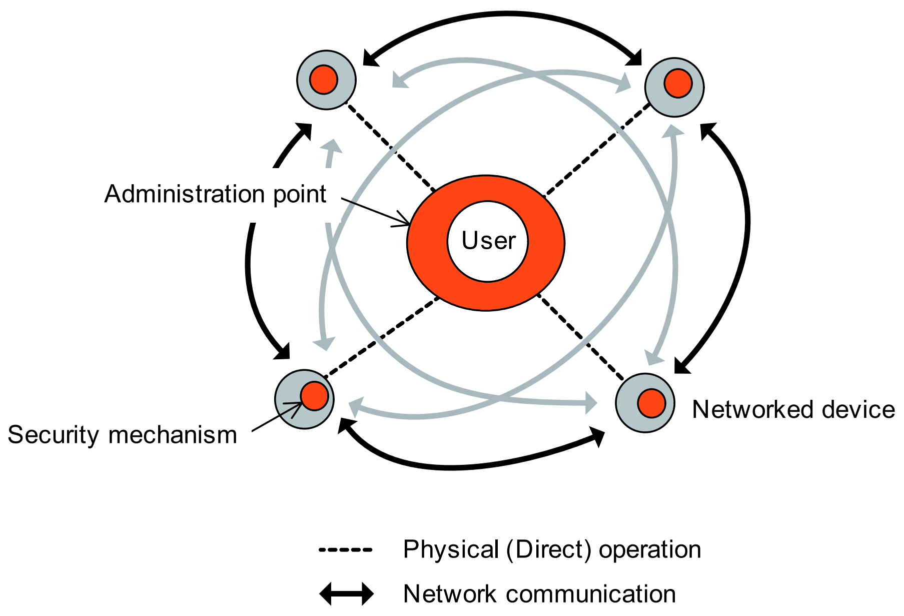


Figure 2. New paradigm for inter-device communication

architecture of the inter-device communication system. For example, this kind of networked device includes future vehicles, building automation systems, home appliances, office equipment and plant machinery.

Future networks will consist of both the inter-device communication paradigm and the conventional *client-server* communication paradigm. Figure 3 shows a mixed network with both paradigms. Figure 3 includes inter-device communication between different administrative domains and communication between the networked device and the conventional server computer.

The term "*device*" in this dissertation means an abstract object with a gateway interface to the Internet. A typical *device* includes multiple closed networks or hard wiring systems that cannot connect to the Internet. The user sometimes operates the *device* directly. However, the *device* sometimes operates automatically based on some condition. In such a case, the *device* behaves as an agent of the user.

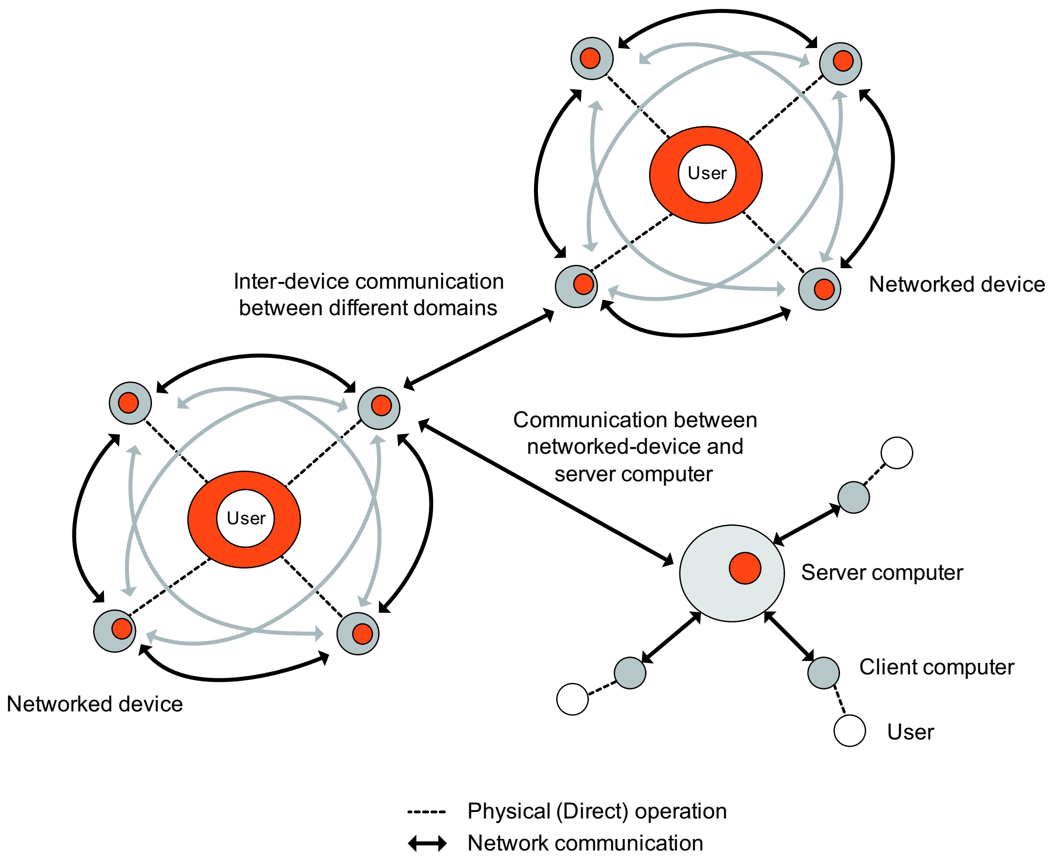

User
User
Networked device
Networked device
Inter-device communication between different domains
Communication between networked-device and server computer
Server computer
Client computer
User
Physical (Direct) operation
Network communication

Figure 3. Mixed network with client-server and inter-device communication

1.3 Features and Security Consideration

This section considers security problems of the inter-device communication paradigm. The inter-device communication paradigm has the following features.

Mixed network architecture of both peer-to-peer and client-server As described section 1.2, future network architecture will be mixed networks with the conventional client-server paradigm and the inter-device communication paradigm. This dissertation considers a novel security mechanism that can be applied to both network architectures.

Single User Multiple Devices (SUMD) These days, a single user can have

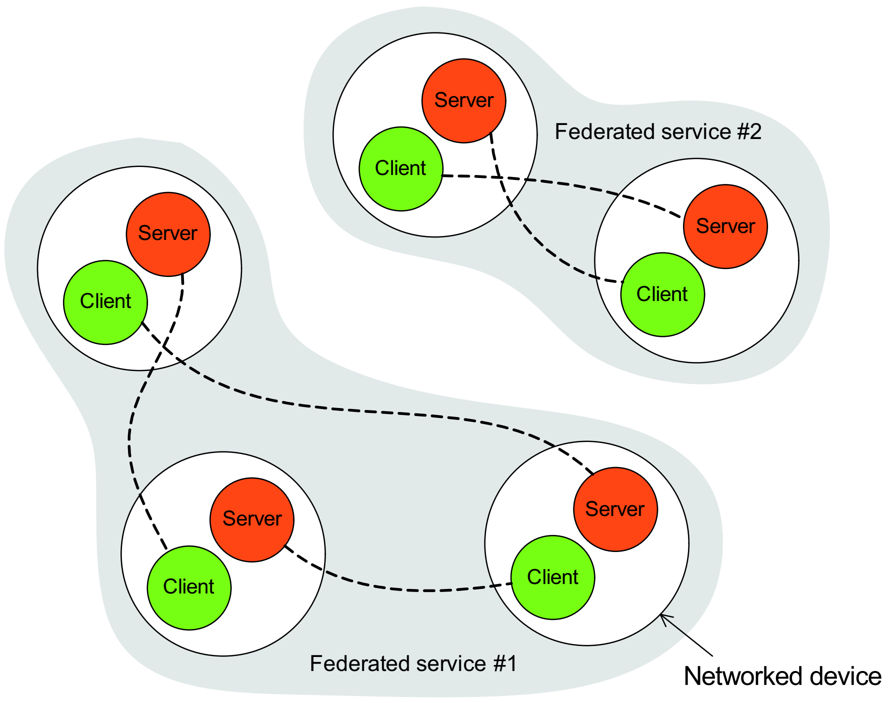


Figure 4. Basic architecture of the inter-device communication system

many electronic devices, such as a mobile phone, PDA, and portable audio player. Such kinds of networked devices continue to proliferate. A user first has to acquire ownership of the device and the device connects to the Internet on behalf of the user. Therefore, we have to handle ownership of devices explicitly. We need a novel ownership management mechanism for such networked devices. We also need a novel collective operation mechanism for such devices such as grouping and domain management. On the other hand, some devices may be used by multiple users. Therefore, it will be more important to distinguish ownership and a device's ID explicitly.

Distributed resources The client-server paradigm has a central administrative server to store each user's data. All confidential data can be stored on a protected central server. However, in an inter-device communication system, an end-point networked device has to protect its own private data and hardware resources. In the client-server paradigm, the administrator has only to protect the central server. In the inter-device communication paradigm, each distributed device has its own private data and hardware resources. The resources on the distributed networked devices have to be protected on each end-point. Therefore, future distributed networked devices have to have a security mechanism on each device. We need a novel device-oriented security mechanism that can be operated on each end-point device. Thus, the inter-device communication paradigm needs a distributed security mechanism.

User-centric administration In the client-server paradigm, administrators in the organization can control software and hardware resources on a server computer. Other users basically do not have to administrate central server resources and they only have to manage each client computer. In the inter-device communication paradigm, each user owns many networked devices other than PCs. Therefore, each end-user has to manage her or his multiple networked devices securely. Each networked device has private data to be managed and access-controlled not by the organization but by each owner. Although the client-server paradigm needs a central administration point on the server computer, the inter-device communication paradigm needs a user-centric administration point for many distributed devices.

Autonomously controlled devices Currently there are many primitive autonomously controlled devices based on simple if-then rules. For example, in a typical inter-vehicle communication system, each vehicle sends its sensor data such as speed and braking data to nearby vehicles. If the vehicle detects unusual dangerous behavior from nearby vehicles, the vehicle will activate its own safety mechanism immediately. Future networks will include many more sophisticated autonomously controlled devices. Autonomously controlled devices are not operated by the owner directly. Therefore, they have to be operated as an agent of the owner. Thus, we need a novel privilege delegation mechanism for such autonomously controlled devices.

For the above features, which differ from the client-server paradigm, conventional security mechanisms cannot be applied to the novel inter-device communication paradigm directly. Therefore, this dissertation proposes a novel security

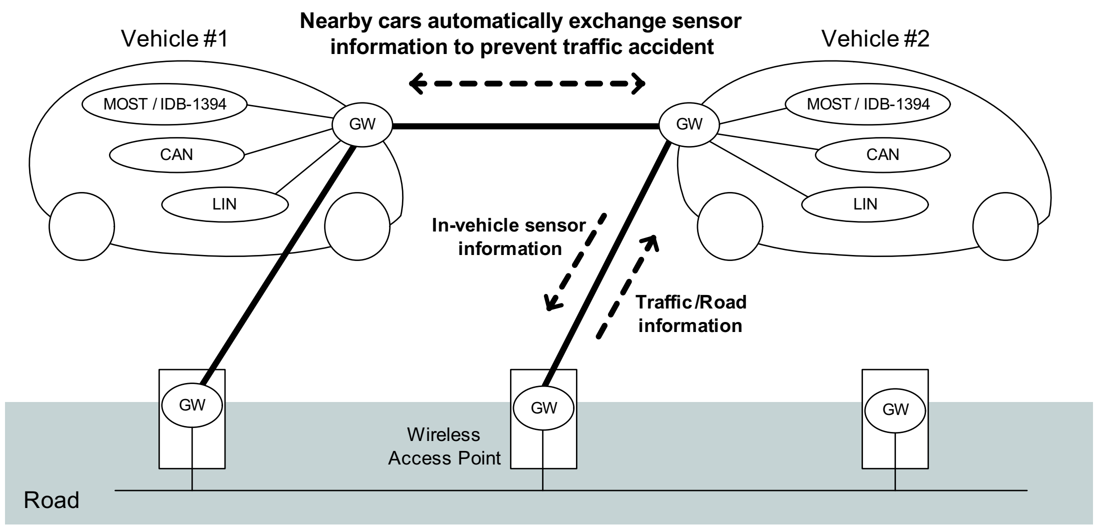

Vehicle #1
Nearby cars automatically exchange sensor information to prevent traffic accident
Vehicle #2
MOST / IDB-1394
CAN
LIN
GW
In-vehicle sensor information
Traffic /Road information
Wireless Access Point
Road

Figure 5. Inter-vehicle and road-to-vehicle communication system

mechanism for the inter-device communication paradigm.

1.4 Industrial Applications based on Inter-device Communication

This section shows some potential industrial applications based on the inter-device communication paradigm and its security considerations. This section intends to show not strict technical details on each industrial application but abstract architecture and practical usage to consider its security mechanism.

1.4.1 Intelligent Transport Systems

A vehicle consists of many electronic parts like sensors and actuators. A vehicle also includes many local area networks between these sensors, actuators and ECUs (Electronic Control Unit). Controller Area Network (CAN) protocol [4] has been widely used in power train control, chassis control and body electronics applications. Almost all ECU suppliers provide a CAN interface. A Local Interconnect Network (LIN) [5] is a less expensive solution. MOST (Multimedia and Control Networking Technology) [6] and IDB-1394 [7] are used as a high-speed network protocol to transfer multimedia data in a vehicle. As described above, current vehicles have many electronic devices and they are interconnected. Because such devices in a vehicle need extremely high reliability and safety, they usually form closed networks.

Currently many inter-vehicle communication systems and road-to-vehicle communication systems are being studied. Figure 5 shows typical inter-vehicle communication and road-to-vehicle communication systems. The vehicles include many networks based on the CAN, LIN and MOST standards. No electronic part is connected to the external network directly. Only a gateway device can be connected to the external network. In a typical system, the vehicle's gateway device automatically exchanges some sensor information with nearby vehicles to prevent traffic accident. The vehicle's gateway device can also communicate to a wireless access point set up on the road. The vehicle also periodically sends useful sensor information from, for example, the velocity sensor, brake sensor and wiper sensor to the wireless access point. These sensor data are organized on a central control center. The reconstructed useful traffic information can be sent to each vehicle from a wireless access point. In 2001, the InternetITS (Internet Intelligent Transport Systems) project [8] showed demonstration experiments using about 1600 networked vehicles in Japan. The DSRC (Dedicated Short Range Communication) [9] [10] protocol has been used since 2001 for the first practical road-to-vehicle system in Japan, the ETC (Electronic Toll Collection System). Many standardization activities for ITS are currently in progress worldwide.

Figure 5 shows how an attacker can cause an accident by sending fake sensor information to nearby vehicles. This kind of attack demonstrates the need for inter-vehicle authentication. Future sophisticated ITS can organize much sensor data from distributed vehicles. For example, a large amount of fake sensor information generated by attackers may cause traffic confusion. To prevent this kind of attack, we need a verification mechanism for legitimacy of each vehicle. The vehicle may also need an access control mechanism for its own sensor data and other privacy information. We also have to consider a novel anonymous authentication method to protect each vehicle's privacy. On the other hand, we also need to check the legitimacy of broadcasted traffic information sent by the wireless access point.

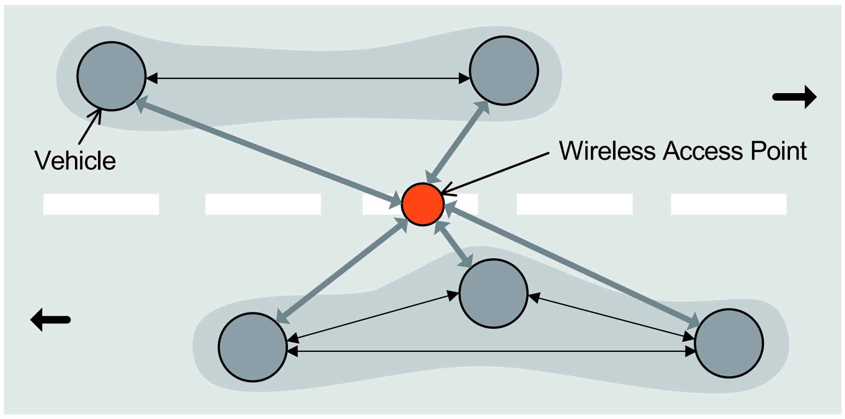


Figure 6. Logical architecture of ITS

A vehicle in a typical ITS has the following features. (1) Inter-vehicle communication forms a peer-to-peer network architecture. Vehicle-to-road communication forms a client-server network. (2) A typical user may have one or two vehicles. However, some organization like a taxi company can have more vehicles. (3) Each vehicle has its own hardware resources and private data to be protected. (4) Each vehicle has to be managed and access-controlled by the owner. (5) A typical vehicle in ITS has some primitive autonomously controlled functions (e.g. an accident prevention system).

Figure 6 shows the logical architecture of ITS described above. This is a mixed network with a client-server paradigm and an inter-device communication paradigm. The communication between vehicles is represented as an inter-device communication paradigm. The communication between each vehicle and the road (wireless access point) is represented as a conventional client-server paradigm. We need an inter-vehicle authentication mechanism and an access control mechanism for in-vehicle information, which can be controlled by a vehicle's owner. These security mechanisms have to be operated on each distributed vehicle.

There is strong competition in the ITS industry worldwide. This dissertation just shows a feasibility study of a novel security mechanism to embed a manufacturer's production ID and a user's ownership into the device in a safe manner. The proposal will be able to show one solution for the authentication and au-

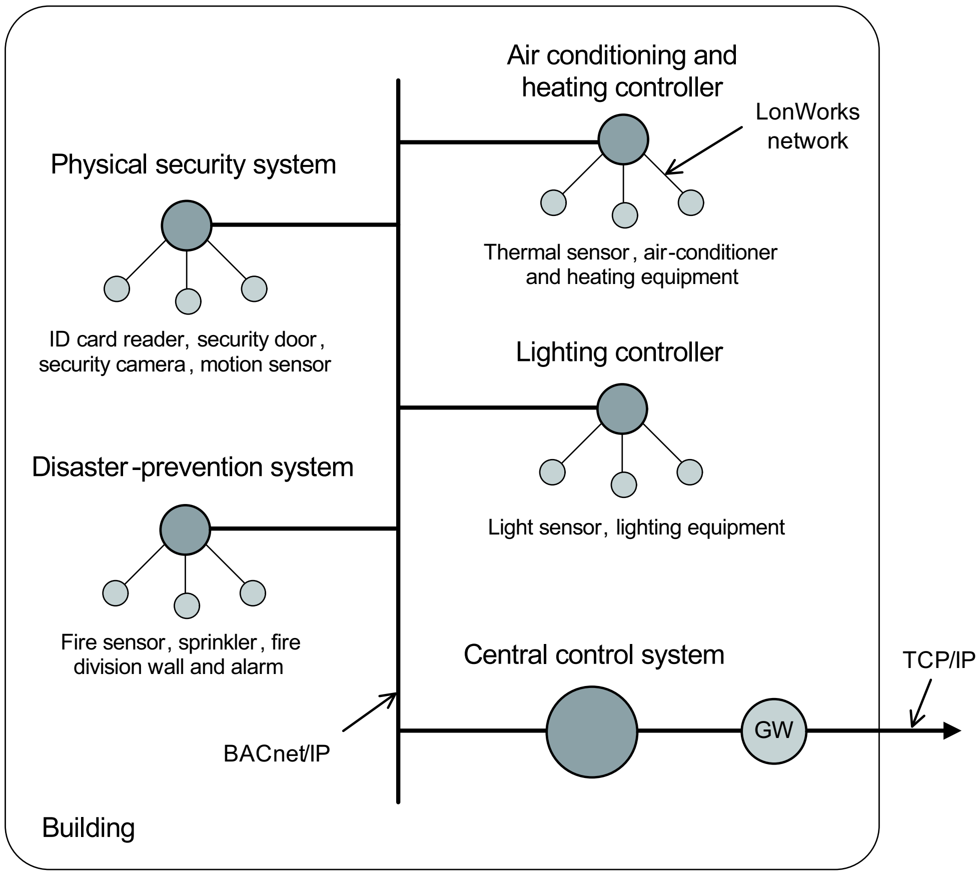

Air conditioning and heating controller
LonWorks network
Physical security system
Thermal sensor, air-conditioner and heating equipment
ID card reader, security door, security camera, motion sensor
Lighting controller
Disaster-prevention system
Light sensor, lighting equipment
Fire sensor, sprinkler, fire division wall and alarm
Central control system
TCP/IP
GW
BACnet/IP
Building

Figure 7. Building automation system

thorization mechanism on the inter-vehicle and vehicle-to-road communication systems.

1.4.2 Building Automation

Figure 7 shows a typical building automation (BA) system. A building has many sensors and electronic equipment. BA systems are typical device-embedded environments and they have the potential to provide future ubiquitous user interface to human beings. Therefore, a security mechanism for future BA systems is important. For example, a typical BA system can control air-conditioners, lighting controllers, heating controllers, physical security systems and disaster-prevention systems, etc. Most controllers in automated buildings employ BACnet (A Data Communication Protocol for Building Automation and Control Networks) [11], BACnet/IP and LonWorks [12] systems. Current typical BA systems consist of two kinds of networks, one for management and another for control. Each network is basically separated and cannot connect to external networks. The central control system collects individual distributed sensor information and can operate each controller appropriately. The building shown in Figure 7 has a gateway to an external network. This gateway may be connected to an external control center or another BA system. The current typical BA system usually forms a closed local area network, and only the central control system can access each controller directly. Current typical BA systems basically form a client-server architecture. Okabe et al. [13] show a specialized security mechanism for a BA system using IPsec with a Kerberos-based authentication system. They propose the application for IPv6 on BA systems. They employ a centralized authentication server approach because they think that public key operation is still expensive on current embedded hardware in BA systems. This dissertation shows one solution for expensive public key operation by using a smart card device.

For example, a local seismograph or a fire alarm device can operate nearby disaster-prevention controllers like fire division walls or sprinklers automatically. If the central control system is down or the network path is broken, this kind of self-defensive architecture is effective. If an automated security camera detects a problem, the camera can operate a nearby security-door device automatically. The above scenarios are based on the inter-device communication paradigm. On the other hand, the following scenarios are based on the conventional client-server paradigm. Emergency earthquake information distributed by the government can be used to execute disaster-prevention systems automatically. An electric power company may optimize power consumption on each automated building by agreement.

If an attacker can send fake information to each controller and the central control system, the attacker can confuse the physical BA system. To prevent such an attack, the central control system and each controller have to authenticate each other. Each controller also has to authorize the access based on the connecting device's identity.

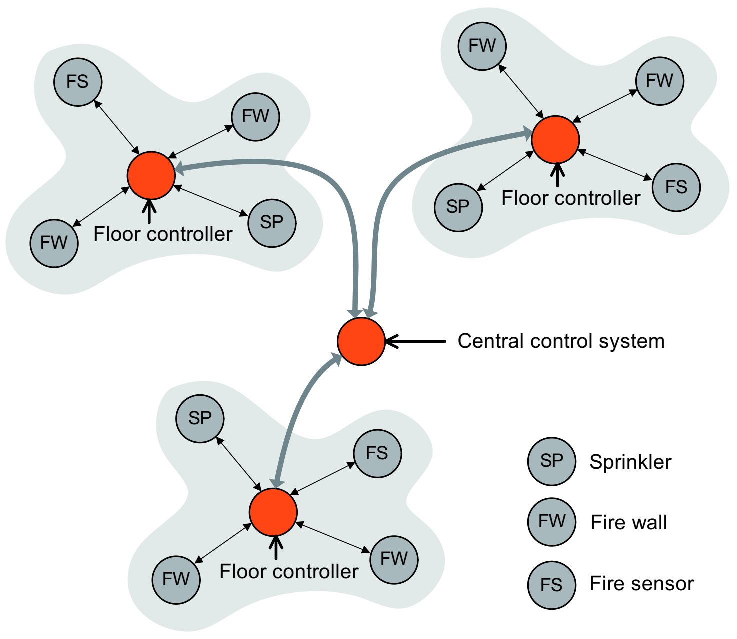


Figure 8. Logical architecture of typical BA system

A typical controller or sensor in a BA system has the following features. (1) The building owner or the security company manages multiple controllers and sensors. (2) Each controller or sensor has its own hardware resources and private data. (3) Each controller or sensor has to be managed and access-controlled by the building owner or the security company, (4) A typical controller or sensor makes a closed physical protected network in the building. (5) A typical controller or sensor has some primitive autonomously controlled functions (e.g. a fire sensor, a disaster prevention system).

Figure 8 shows the logical architecture of a typical BA system. Figure 8 includes three independent closed networks with each floor's main controller. Each network connects to the central control system. In each closed network, devices will interact with each other. Thus, a typical BA system includes both a client-server paradigm and an inter-device communication paradigm. Future sophisticated BA systems will be more distributed and autonomously federated.

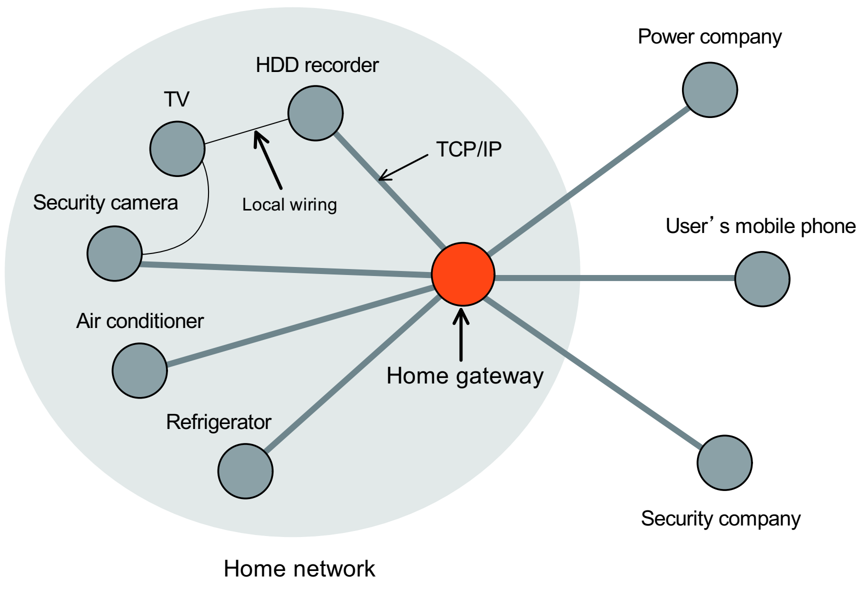


Figure 9. Logical architecture of typical home networking system

1.4.3 Home Networking

The architecture of home networks is similar to a BA system. Figure 9 shows the logical architecture of a home networking system. In this example, some appliances are connected to each other by a local wiring system. Appliances in the home network connect to the home gateway, which can connect to the Internet. The home gateway behaves as a proxy for the appliances. The inhabitant can control the appliances via mobile phone or other networked device from outside the home. The security camera can be accessed by a hired security company. The power company can control power consumption of some appliances in the home by agreement. As described above, some home appliances can be controlled by the inhabitant or other authorized organizations. The home gateway is a key component of home networking technology. However, the home gateway has the limitation to control each resource on each home appliance. If some appliances employ wireless communication then an attacker can attack such end-point devices. Therefore, each home appliance has to be equipped with its own security mechanism. Currently there are many specifications for home networking

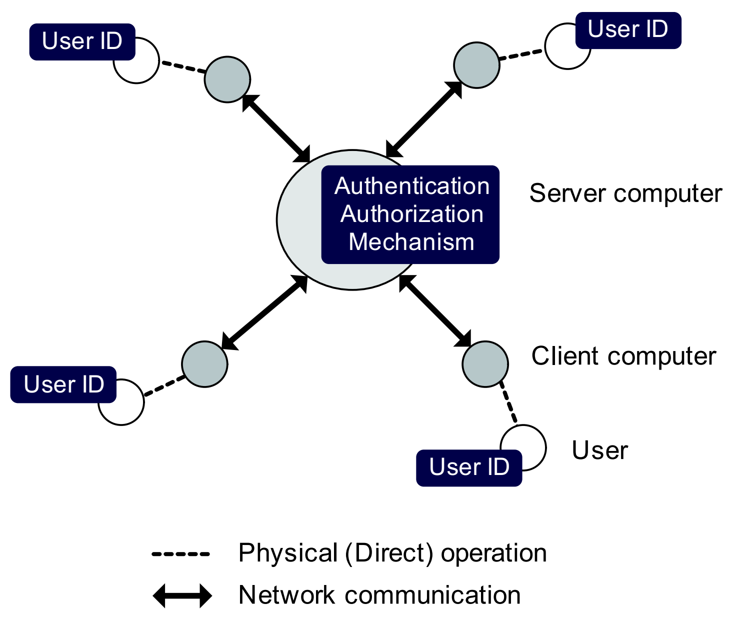


Figure 10. Relation between IDs and security mechanism in the client-server paradigm

like ECHONET [14], UPnP [15] and DLNA [16]. In most home networking systems, wired or wireless Ethernet, PLC (Power Line Communication) are used for physical communication media.

1.5 Requirement Analysis for Security Mechanism on Inter-device Communication Paradigm

This section discusses the security mechanism on the inter-device communication paradigm. Figure 10 shows the relation between user ID and security mechanism in a conventional client-server paradigm. Each user has its own user ID, which is managed by the user. Typical user authentication is processed by a pair of user ID and password, a key-pair based on a public key algorithm, biometrics and their combination. Users are authenticated on a central server computer using her or his user ID and secret information such as a password. Therefore, the server computer has only to equip the authentication and authorization mechanism.

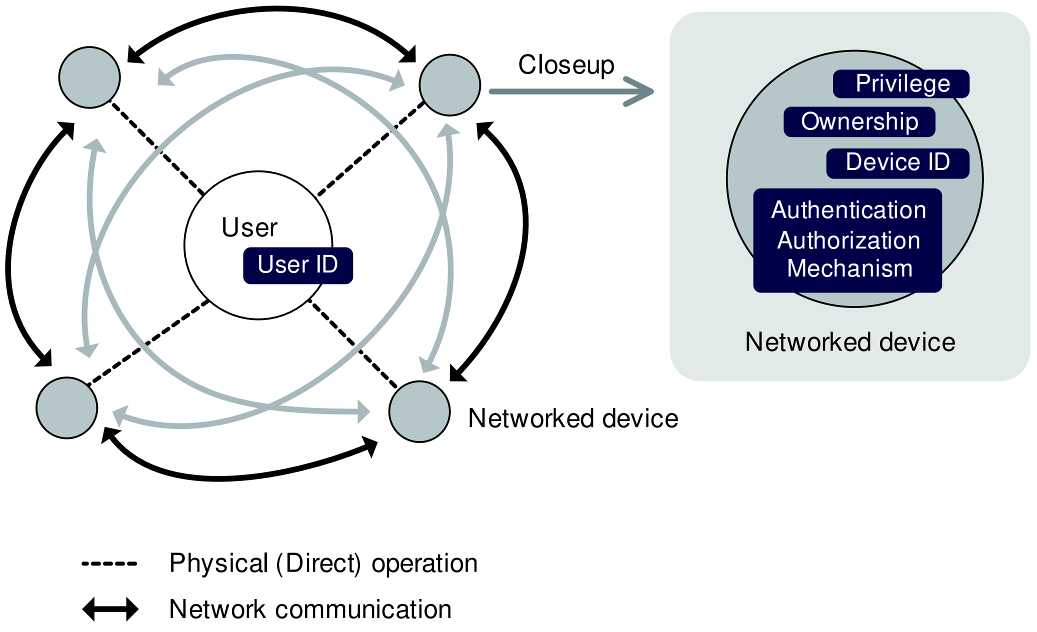


Figure 11. Relation between IDs and security mechanism in inter-device communication paradigm

In typical client-server systems, the purpose is just to provide network services to end-users. Therefore, the service is not dependent on the hardware. Most client-sever systems like the WWW and e-mail essentially can work on any hardware. Next, Figure 11 shows the relation between IDs and security mechanism in the inter-device communication paradigm. In the inter-device communication paradigm, a single user can have multiple networked devices.

This section shows some requirement analysis for a security mechanism for the inter-device communication paradigm as follows.

Authentication and authorization in distributed environment Although the client-server paradigm needs a centralized authentication and authorization mechanism, the inter-device communication paradigm essentially needs distributed authentication and authorization mechanisms. Typical networked device sometimes makes an ad-hoc network without using a central administrative server. A pair of user ID and password is the most-used local authentication mechanism. In the typical pre-shared key mechanism, as the number of devices connected to each other increases, the number of passwords causes combinatorial explosion. Therefore, the user would not administrate her or his many devices. If all devices have the same password then the administration is easier for owners. However, it only provides limited security and it is essentially insecure. Sometimes this pre-shared secret mechanism employs an external central authentication server. If the user employs this kind of external authentication server, someone has to maintain the server machine and the use of the device is restricted by its location. If the device moves to another location where it cannot connect to the authentication server, the device will not communicate at all. Kerberos [17][18] is typical single sign-on system on the Internet. The Kerberos system consists of a central authentication server and ticket-granting server. Kerberos is a partially centralized security mechanism. The user needs authentication from the server before login to the domain. After the user is authenticated, the user does not need the central authentication server. Administrators have to manage these Kerberos servers and user accounts. Kerberos has the same problems like the central authentication server described before. Although the location problem may be resolved by mobility-supported protocols, the administration cost is still a problem. PKI (Public Key Infrastructure) [19][20] can provide a distributed authentication mechanism without an online central server. PKI has some problems such as time-lag of certificate revocation and complexity of administration. However, PKI has the potential to provide a distributed authentication mechanism on the Internet.

Separation between device's ID and other attributes The client-server system provides only software services and it is not dependent on the hardware itself. However, in the inter-device communication paradigm, many manufacturers have released a wide variety of devices. Each device has to identify each device's vender ID, model number, production number and other information about interoperability correctly. Therefore, each device has to validate whether the target device provides a trusted environment or not. The need for the validation mechanism indicates the separation of a device's

ID and other attributes like ownerships and privileges. The user has multiple devices and these devices have to be managed by the user. Therefore, the ownership of the device will be the most important attribute on the device. The other attributes like special privileges purchased from some service provider are also important for practical use. Such kinds of special privileges can be used for a service that needs an accounting mechanism. The dissertation stresses the importance of separation between a device's ID and other attributes on the device. This separation means that the manufacturer manages the device's ID and the end-user manages other attributes like ownership. The device's ID is usually persistent data. Other attributes like ownership of the device are not persistent and can be changed by the owner.

Association between device's ID and attributes As described above, the dissertation proposes separation between a device's ID and other attributes. However if a device's ID and other attributes are completely separated, the system cannot work well. Their attributes have to be valid only on specified device and not on another device. If other attributes can be valid on another device then it causes a risk of misuse. Therefore, we have to guarantee the relation between the device's ID and its attributes like ownership.

ID protection on the distributed device The devices are distributed and sometimes become physically insecure. For example, the home network is usually located in a physically protected house. However, mobile devices that can be moved outside become insecure and can be accessed by an attacker physically. A vehicle is protected by its own physical security mechanism. Controllers and sensors in a BA system can be accessed by an attacker physically if the building lacks physical security. As described above, the distributed devices always face a risk of physical attack. If the device is stolen then the ID and attributes on the device will be abused. Therefore, we need a novel ID protection mechanism for distributed devices. The device's ID and other valuable attributes must be protected from physical attack. A tamper-resistant device like a smart card is well-suited for such a security mechanism. Moreover, each tamper-resistant device has to be

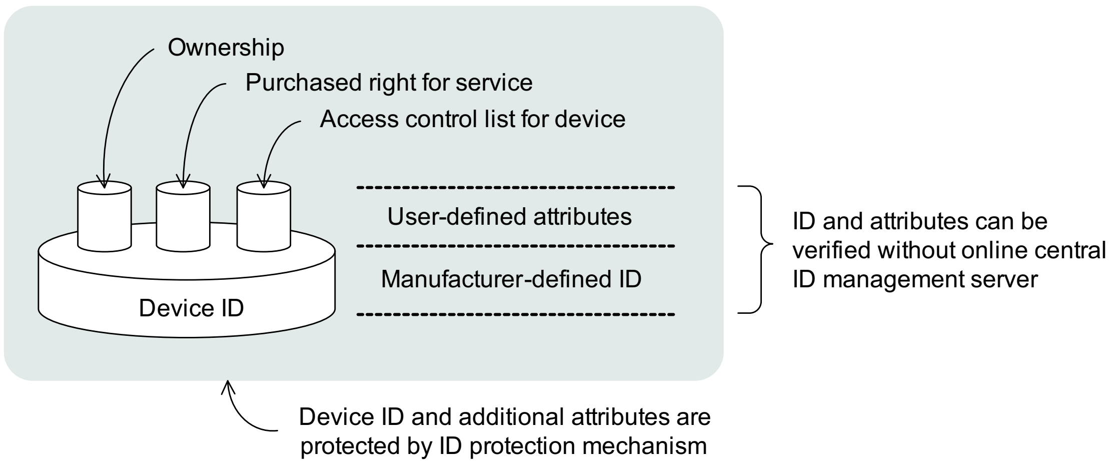

Ownership
Purchased right for service
Access control list for device
User-defined attributes
Manufacturer-defined ID
Device ID
ID and attributes can be verified without online central ID management server
Device ID and additional attributes are protected by ID protection mechanism

Figure 12. Security model for ID management on inter-device communication paradigm

protected by each owner's secret information (e.g. PIN code, biometrics).

Independence from applications and implementations The security mechanism for inter-device communication has to be interoperable. There are many protocols and specifications on each industrial arena like ITS, BA and home networks. Therefore, it is extremely difficult to propose a versatile security mechanism for these fields in the real sense. The dissertation proposes a novel security mechanism for inter-device authentication and authorization. The security mechanism must not be dependent on each application and implementation.

As described above, the inter-device communication paradigm has different characteristics from the client-server paradigm. Therefore, we need a novel ID management mechanism for these distributed networked devices. This dissertation proposes that a device has two kinds of rights. The first right is a device's ID that is used to verify the legitimacy of the device. The device's ID is issued by manufacturers. The second right is additional attributes for the device like ownership. These additional rights are mainly used to control the access to the device. Figure 12 shows a security model for the ID management the dissertation proposes. We need a novel security mechanism to embed a device's ID and additional attributes like ownership to the device in a safe manner. To deploy in a practical environment, we also need an inter-device authentication protocol and encrypted communication protocol between two devices. On the other hand, the inter-device communication paradigm needs a novel user-oriented administration mechanism. We need a user-friendly tool to manage the device's ID and attributes. The inter-device communication paradigm will also allow autonomously controlled devices. For such devices, we need some sort of privilege delegation mechanism for user-managed devices.

1.6 Federation of Networked Devices and Resources

As described in section 1.4, actual industrial applications employ each specific protocol and framework. However, the dissertation assumes these inter-device communication systems have common characteristics. This section shows some technical building blocks that have the potential to federate devices in the inter-device communication paradigm. The dissertation discusses these actually used frameworks and considers their common characteristics for federating future networked devices.

1.6.1 Web Service Based Federation

The concept of federating services is a topic of interest in the Web engineering community [21] [22]. W3C (World Wide Web Consortium) defines the Web service as follows [23]:

> A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

Figure 13 shows the general process of Web service. The developer first implements the service on the *provider agent*. The service description is expressed by

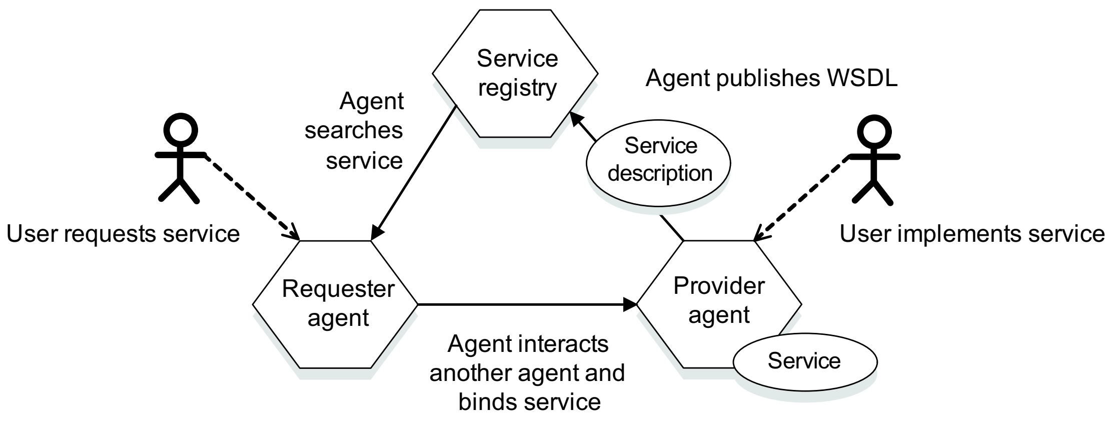


Figure 13. General process of Web service

the WSDL (Web Service Description Language). The service description is managed by the *service registry*. The *requester agent* finds the service and interacts with the *provider agent*.

Gaedke et al. propose the FDX (Federated Device Assembly) architecture [24]. Although most Web service provides federating service, they introduce the concept of federated devices based on conventional Web service architecture. Igaki et al. also show the federation mechanism for home appliances based on Web service [25]. Figure 14 shows basic architecture of Web service. Web service technology is based on conventional communication protocols like HTTP [26] and TCP/IP. SOAP (Simple Object Access Protocol) [27] is used for XML-based messaging service.

1.6.2 UPnP-based Federation

A UPnP (Universal Plug and Play) [15] system basically consists of *device*, *service*, and *control point*. Figure 15 shows basic components on UPnP system. The *device* is a container of *services* and nested *devices*. The *device* has an XML-based *device description* document. The *service* also has an XML-based *service description* document. The *service* has state variables, the control server and the event server. The *control point* is a controller of the *devices*. The *control point* discovers *devices*, retrieves *device descriptions* and *service description*, invokes

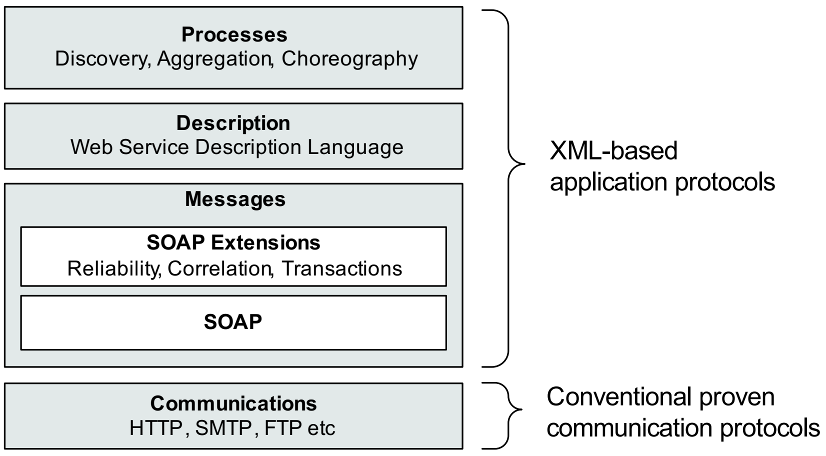

Processes
Discovery, Aggregation, Choreography
Description
Web Service Description Language
Messages
SOAP Extensions
Reliability, Correlation, Transactions
SOAP
Communications
HTTP, SMTP, FTP etc
XML-based application protocols
Conventional proven communication protocols

Figure 14. Basic architecture of Web service

the actions and subscribes to the service's *events*.

UPnP framework mainly provides the following functions: (1) addressing, (2) discovery, (3) description, (4) control, (5) eventing and (6) presentation. UPnP framework employs DHCP (Dynamic Host Configuration Protocol) [28] or Auto IP mechanism [29] for the addressing of the *device*. The *device* attached to UPnP network can advertise its *services* to the *control points*. The *control point* first searches for the *devices* in UPnP network. The *control point* knows about the *device* after the discovery. The *control point* retrieves the *device description* document from the URL providing the discovery message. The *device description* document written in XML includes the serial number, the model name, the manufacturer's information and the URL for the *service description* document. The *control point* also retrieves the *service description* document written in XML. The *control point* executes the device's function to send the XML-based control message to the appropriate URL. The *service* has some state variables. When the value of the variable is changed, the *control point* can receive the *event* message. If the *device* has the presentation function, the user can browse the device's web pages to control it.

UPnP framework is constructed on a conventional TCP/IP stack and existing application protocols. Figure 16 shows layer model of UPnP framework. The

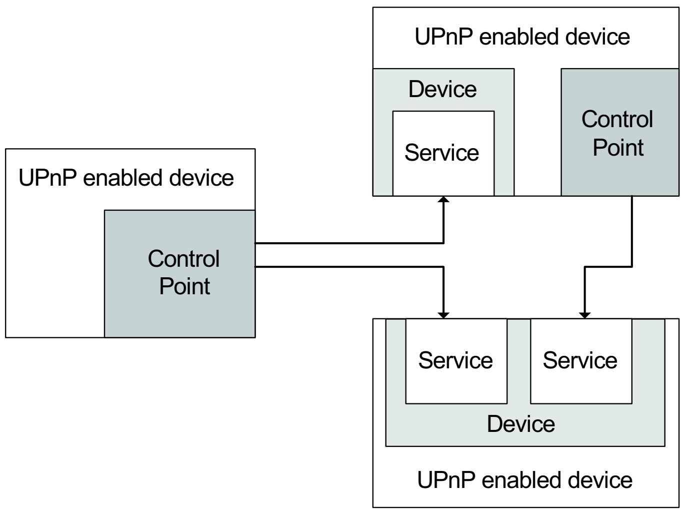


Figure 15. Basic components on UPnP system

addressing is performed by DHCP or Auto IP as described above. The advertisement and search for the device are performed by SSDP (Simple Service Discovery Protocol) and GENA (Generic Event Notification Architecture) over HTTPMU (Hypertext Transfer Protocol Multicast over UDP)/HTTPU (Hypertext Transfer Protocol Unicast over UDP). SSDP, GENA, HTTPMU and HTTPU are expired Internet Drafts in the IETF. The description is simply performed by HTTP's GET method [26]. The control is performed by SOAP over HTTP. The eventing is performed by GENA and HTTP.

In summary, UPnP framework has fundamental functions to handle networked devices, such as addressing, discovery, description, control and event. UPnP framework clearly defines *device*, *service* and *control point* entity. UPnP framework is constructed on the standardized protocols.

1.6.3 Jini-based Federation

The Jini architecture [30] also provides a fundamental framework for networked devices. The Jini architecture is constructed on the Java virtual machine environment. Figure 17 shows the basic components on the Jini network. The Jini consists of the following entities: the *client*, the *lookup service* and the *service*

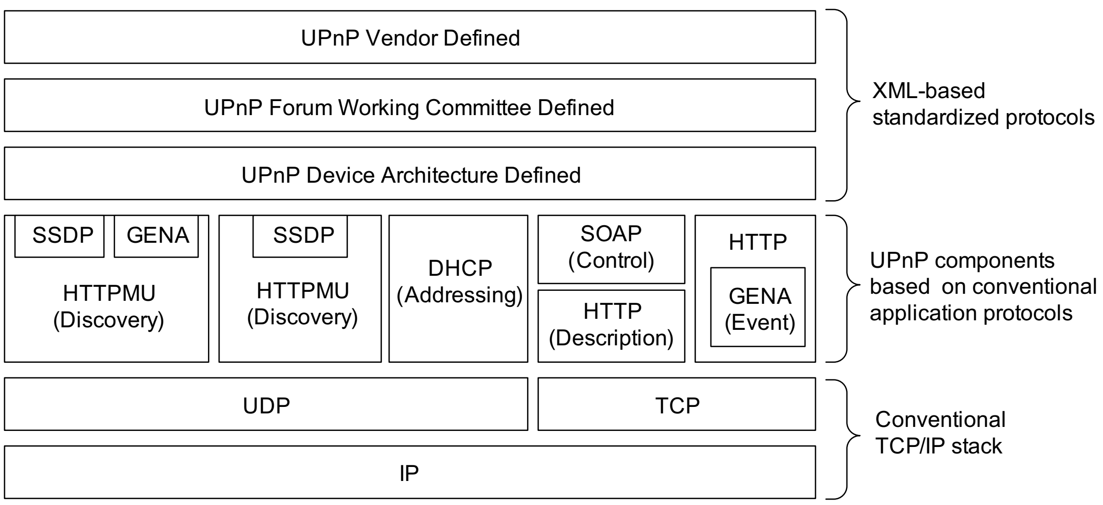

UPnP Vendor Defined
UPnP Forum Working Committee Defined
UPnP Device Architecture Defined
XML-based standardized protocols
SSDP
GENA
HTTPMU (Discovery)
SSDP
HTTPMU (Discovery)
DHCP (Addressing)
SOAP (Control)
HTTP (Description)
HTTP
GENA (Event)
UPnP components based on conventional application protocols
UDP
TCP
IP
Conventional TCP/IP stack

Figure 16. Layer model of UPnP framework

provider. The *client* is the device used by the user or another device. The *service provider* is the device that provides actual services. In the first step, the *service provider* registers its services to the *lookup service*. The registered objects include the *service object* that can work on the Java VM. The *client* downloads the *service object* from the *lookup service*. The *client* runs the *service object* to execute the remote device's actual service. The *service object* and the *control object* communicate using Java RMI (Remote Method Invocation). The *service object* is a mobile code, it moves from one machine to another machine.

The Jini architecture consists of the following functions: (1) discovery, (2) join, (3) lookup, (4) Java RMI, (5) leasing, (6) transaction, (7) event and (8) security. At first, the *service provider* searches the *lookup service* in the Jini network using the discovery function, and registers its services using the join function. The *client* searches for the appropriate service from the *lookup service* and executes using the lookup function. The actual code is executed by Java RMI. The Java RMI is similar to the RPC (Remote Procedure Call) mechanism. The remote code (Service Object) can execute another machine's code (Control Object) as a local code. The leasing defines the time to lease each service. The *service provider* can notify the *client* of the event. The Jini also employs two-phase commitment

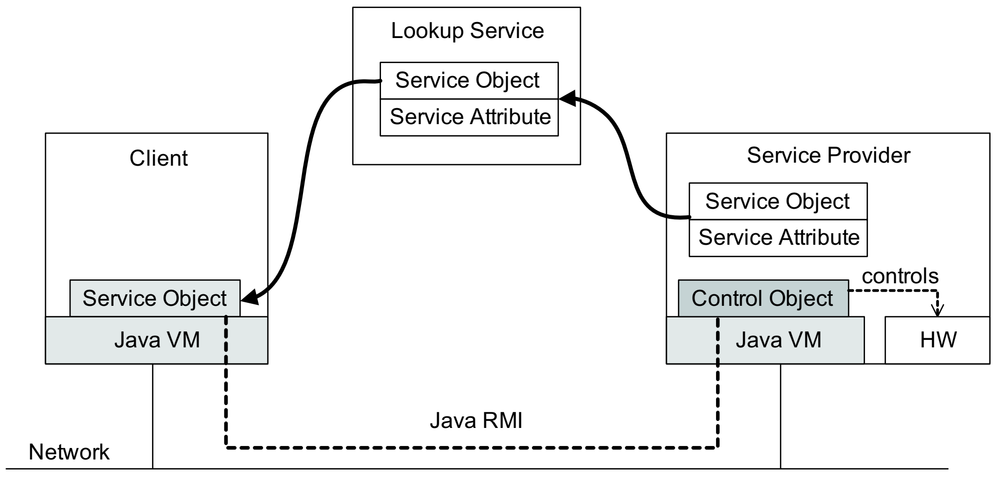


Figure 17. Basic components on Jini network

for its transaction mechanism.

1.6.4 Other Frameworks

There are some other industrial proposals to manage networked devices. For example, DLNA (Digital Living Network Alliance) [16] is a practical application of the UPnP framework. HAVi (Home Audio/Video interoperability) [31] was the standard for the IEEE1394 networked devices. ECHONET [14], proposed by Japanese companies, also provides a framework for home appliances. CORBA (Common Object Request Broker Architecture) [32] is the fundamental framework for distributed object technology. Microsoft also provides *.NET remoting framework* [33] to execute remote procedure calls on Microsoft Windows. BACnet [11] and BACnet/IP and LonWorks [12] are open networking technology for building automation. They are used for lighting, air conditioner, alarm, security system, etc. They usually have a closed network in the building. DSRC [9] [10] is the Japanese proposed standard for wireless communication in ITS.

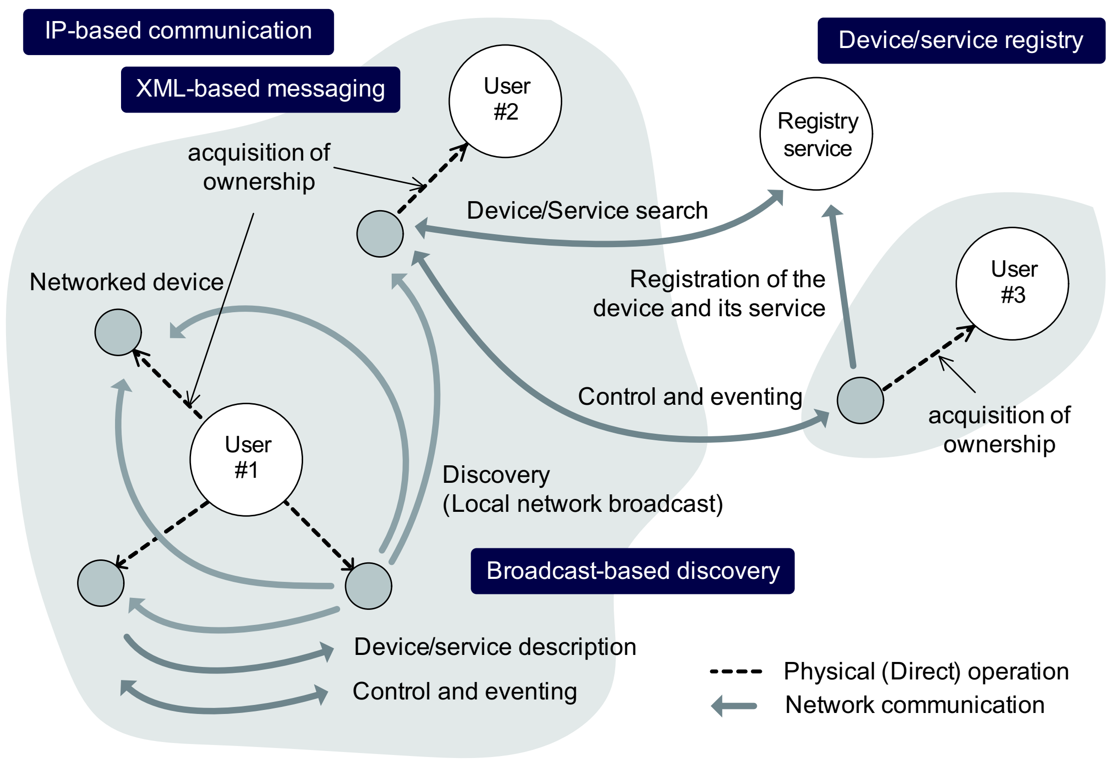


Figure 18. Abstract federation model

1.6.5 Abstract Federation Model

As mentioned above, there are many existing frameworks and specifications that have the potential to federate devices. Each framework basically provides common characteristics such as discovery, service description, control and eventing. The UPnP framework employs a service discovery mechanism as network broadcast. Web service and Jini employ centralized registry and look-up service. Figure 18 shows the abstract federation model the dissertation assumes. UPnP and Web service provide service description and control mechanisms based on XML messages. XML is a well-suited method to provide interoperable communication between heterogeneous frameworks and specifications. UPnP and Web service are basically constructed on conventional network systems like TCP/IP. Jini just provides an RPC mechanism on both Java virtual machines. Each framework considers the interoperability problem by each method. On the other hand, BACnet, LonWorks and DSRC basically provide closed network architecture only. These specific protocols can only used with homogeneous systems.

1.7 Existing Security Mechanisms

Web Service Security [34] proposed by OASIS (Organization for the Advancement of Structured Information Standards) is based on XML-Signature and XML-encryption. The Web Service Security can employ Kerberos [17] and PKI (Public Key Infrastructure) [35][19] to authenticate user. SAML (Security Assertion Markup Language) [36], which supports Single Sign On, is also used in Web Service Security. The security mechanism of the Web service also employs XACML (Extensible Access Control Markup Language) [37]. Most security mechanisms of the Web service are based on XML. Thus, Web Service Security is basically constructed on the application-level.

UPnP Security [38] is an extension of the conventional UPnP framework. UPnP Security handles a device's identity as *SecurityID* (the hash of the public key). UPnP Security can handle the raw *public key*, not the *public key certificate*. The template of the access control rules for each UPnP-aware device is provided by the manufacturer. The user can define her or his access control rules on the devices. UPnP messages are protected by the encrypted SOAP messages. UPnP Security is completely an application layer's security mechanism. Therefore, UPnP Security can only apply the UPnP framework. It cannot apply another framework for networked devices.

The security mechanism of the Jini is based on Java technology. Therefore, the mobile *service object* is protected by the code signing, the code verifier and the access control mechanism. The Jini architecture is an extension of the Java technology. The Jini architecture itself does not provide the authentication, authorization and traffic confidentiality mechanisms between the hosts. BACnet is usually used in the physical protected network in the building. BACnet has a limited security mechanism and basically assumes all devices are trusted. DSRC also defines its own security mechanism such as an authentication mechanism and a key exchange mechanism. The standardization processes for ITS are currently in progress worldwide.

As mentioned above, there are many security mechanisms for each framework and specification. However, these security mechanisms cannot meet the requirements shown in section 1.5.

1.8 Target Framework for Security Extension

Section 1.6 shows some potential frameworks and the abstract federation model. Section 1.7 shows some existing security mechanisms for each framework. This dissertation aims to propose a security model that is not dependent on each framework and specification. This dissertation assumes that the target framework has the following technical features.

- The framework provides basic functions to federate devices, such as discovery, service description, control and eventing. These functions are provided in an application layer's protocol.
- Each device has both client and server functions. Thus, devices can make both peer-to-peer and client-server architecture.
- The communication function is provided by conventional TCP/IP protocol stack.
- The framework's security mechanism is provided as an extension of the application layer's protocol. The framework's own security mechanism can be disabled by its configuration as needed.

Section 1.5 shows requirement analysis of a security mechanism for the inter-device communication paradigm. The dissertation proposes a novel security mechanism that meets the requirements and can apply to the above abstract framework.

1.9 Security Basis and Scope of the Dissertation

From the perspective of the network security mechanism, the OSI reference model describes the following five classes of *security services* [39] [40]: (1) Authentication, (2) Data confidentiality, (3) Data integrity, (4) Access control, and (5)

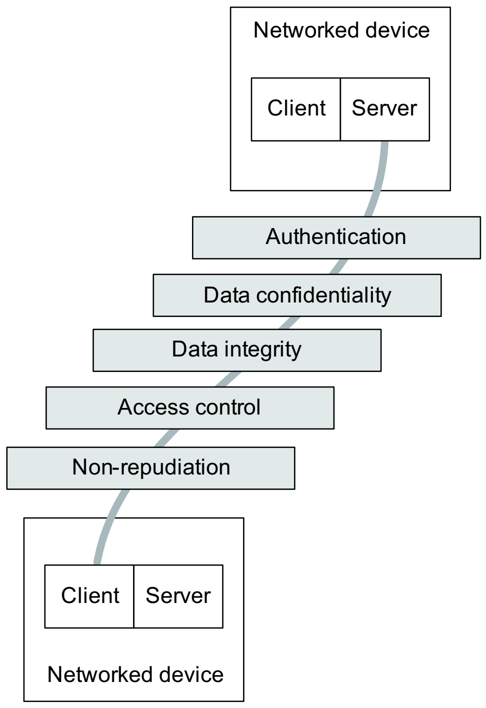


Figure 19. Basic security services described by OSI reference model

Non-repudiation. The protocols used on the inter-device communication must satisfy the above requirements. Figure 19 shows the basic *security services*. The OSI security architecture also shows a couple of the *security mechanisms* that can be used on the above requirements. These *security mechanisms* are: (1) Encipherment, (2) Digital signature mechanism, (3) Access control mechanism, (4) Data integrity mechanism, (5) Authentication exchange mechanism, (6) Traffic padding mechanism, (7) Routing control mechanism, and (8) Notarization mechanism. Complementary to these security mechanisms, the OSI security architecture also shows the following *pervasive security mechanisms*: (1) Trusted functionality, (2) Security labels, (3) Event detection, (4) Security audit and trail, and (5) Security recovery.

On the other hand, from the perspective of the host security mechanism, the NCSC (National Computer Security Center) of the U.S. National Security Agency (NSA) developed the Trusted Computer Security Evaluation Criteria (TCSEC), also known as the "Orange Book" in the late 1980s [41]. The Orange Book defines the following security requirements for a trusted computer system: (1) Security policy, (2) Marking (Access control labels associated with objects), (3) Identification, (4) Accountability (Audit logs), (5) Assurance, and (6) Continuous protection. The Orange Book defines the seven sets of evaluation criteria called classes (D, C1, C2, B1, B2, B3 and A1) based on the above requirements. The ITSEC (Information Technology Security Evaluation Criteria) [42] proposed in Europe also defines its own sets of evaluation criteria. The ISO/IEC 15408 is the current common criteria [43] for international standards on this research area.

This dissertation focuses on the security mechanisms for inter-device communication. Therefore, the host security mechanisms are outside the scope of this dissertation. For securing inter-device communication systems, the dissertation especially focuses on the following security requirements: authentication, access control, data confidentiality and integrity for inter-device communication.

1.10 The Dissertation's Contribution

For securing an inter-device communication system, this dissertation presents the concept of a novel security mechanism for inter-device authentication and authorization. The dissertation defines the *device's identity* and the *ownership information* (or other additional attributes) separately. In the proposed inter-device authentication and authorization framework, The *manufacturer* can install the *device's identity* on the *device* in the production phase. After purchasing the product, the *user* can transfer her or his *ownership* and access control list to the *devices*. The *device's identity* and the *ownership* are strongly bound by the PKI technique. In the operational phase, the *device* first executes the inter-device authentication using each *device's identity*. Next, the *devices* can enforce the access control rule based on the *device's identity* and the peer *ownership*.

This dissertation also proposes a method to store the *device's identity* and *ownerships* on the *device* securely. Therefore, the dissertation proposes novel smart card software for the device's security. The dissertation integrates the proposed framework into the proven network-layer's security mechanism, especially IPsec protocol and IKEv1/IKEv2 protocol. As a result, this dissertation shows a feasibility study of the proposed device-oriented ID management model by the prototype implementation. In brief, the overall security mechanism in the prototype implementation provides inter-device authentication and authorization, a protection method for the device's identity, traffic confidentiality and integrity.

1.11 Organization

The rest of the dissertation is organized as follows. Chapter 2 describes related work on inter-device communication. Chapter 3 briefly introduces the dissertation's main challenges. Chapter 4 shows how to apply a host-based security mechanism to user-level applications. Chapter 5 presents the inter-device authentication and authorization framework guaranteeing explicit ownership. This chapter presents the main concept of the proposal, the design and the prototype implementation. Chapter 6 presents the demonstration experiments to show the usability of the proposed inter-device authentication and authorization framework. Chapter 7 provides some discussions about life cycle management for devices, the hardware dependency problem, group management methods for devices, prospective future applications, bridging architecture among heterogeneous systems and future work. Chapter 8 concludes with a summary.

2. Related Work

This chapter shows related work on the inter-device communication paradigm and makes clear the dissertation's scope in the research area.

2.1 Trust Model

Josang et al. show trust requirements in ID management [44]. They first define entities, identities and identifiers for a general ID management system. They clearly define the name space for a specific domain and an identifier provider and users in the domain. The identifier provider has to provide: (1) a method for protecting client privacy, and (2) user registration procedures and an authentication mechanism. They also show a federated ID management model across different domains. The federated ID management model is used for SSO (Single Sign On). They also show another model, a centralized ID management model like PKI. Pathare et al. show a security mechanism for an ad-hoc network [45]. They state a Single User Multiple Device (SUMD) paradigm. A user holds more than one mobile device. They state that it is difficult to handle the multiple device identities. Therefore, they employ a combined identity for a user and a device. Their proposal is based on the ECC (Elliptic Curve Cryptography) algorithm. Burnside et al. describe a resource discovery and communication system [46]. They take a proxy-based approach. They show two important types of protocols: a protocol for secure device-to-proxy communication and a protocol for secure proxy-to-proxy communication. They employ a lightweight protocol for device-to-proxy communication. They also employ SPKI/SDSI (Simple Public Key Infrastructure/Simple Distributed Security Infrastructure) [47][48] for proxy-to-proxy communication. Erber et al. define a pattern system for AAIs (Authentication and Authorization Infrastructures) [49]. They state the importance of pattern systems using open standard languages like SAML [36] and XACML [37].

2.2 Device's Domain and Group Management

Sailer et al. propose a pervasive authentication domain for automatic device authorization [50]. They state the importance of a grouping mechanism for pervasive mobile devices and propose a central personal authorization gateway. Each authenticated device behaves as an authenticated user in the gateway. Zou et al. show an authentication and authorization mechanism for group communication in grid computing [51]. They employ a Virtual Organization level for grouping, Role-Based Access Control (RBAC) [52] and an attribute-based approach. Capkun et al. propose the "integrity regions" [53]. Example applications of integrity regions are: (1) key establishment for Diffie-Hellman public keys between two users, and (2) device authentication between a user and a device. They employ distance information (small physical space) in the integrity regions.

2.3 Device and Service Discovery

Haque et al. propose an authentication-based lightweight device discovery model for inter-device communication [54]. They employ an LPN (Learning Party with Noise)-based H-C (Human-Computer) authentication protocol. Shacham et al. show architecture for device discovery, device configuration and the transfer of the active sessions between devices. They also show a user-control method of location-based behavior and handling of security and privacy concerns [55]. They assume the SIP (Session Initiation Protocol) as a standardized, widely used signaling protocol for IP-based multimedia services.

2.4 Control Architecture

Weippl et al. propose a personal trusted device for web services [56]. They introduce the concept of multi-level security (MLS) [57] to protect PDA when using web services. Dellutri et al. propose a local authentication system using a Bluetooth-enabled device as a personal trusted device (PTD) [58]. Salvador employs a special device called a Personal Command Module (PCM) [59]. The PCM is a control device for each user. The concept of the PCM is similar to the control point in the UPnP framework. He defines ownership of a device and delegation of a user's capability. His framework can make groups of devices based on the ownership.

2.5 Network and Mobility

In 1989, Bellovin gaves a brief summary of security problems in the TCP/IP protocols [60]. Ferguson et al. showed the cryptographic analysis of IPsec RFCs in 1999 [61]. More recently, there are many works on mobile ad-hoc networks. The IETF discusses MANET (Mobile Ad-hoc Networks) [62], Mobile IP [63][64] and NEMO (Network Mobility) [65]. Ahmed et al. show the binding update authentication scheme for the mobile IPv6 route optimization mechanism [66]. Weimerskirch et al. handle security problems on mobile ad-hoc networks [67]. They state the importance of a security mechanism without any pre-established secret or common security infrastructure like PKI. They propose an efficient zero common-knowledge authentication (ZCK) protocol.

2.6 Trusted Computing

Garfinkel et al. present a virtual machine-based platform for trusted computing [68]. They employ virtual machine monitor technology to protect platform resources. They insert a thin layer between hardware and operating systems to control and encrypt data. A virtual machine monitor requires a powerful processor, however, it is a useful concept for most security mechanisms. Dietrich shows a concept for enhancing mobile and embedded devices with trusted computing technologies [69]. Dietrich employs a "Mobile Trusted Module" in Sun's J2ME platform as a trusted computing base. TCG (Trusted Computing Group) publishes TPM (Trusted Platform Module) specification [70]. TPM is a tamper-resistant device that can be embedded in the computer.

2.7 Secure Storage and Enterprise Data Security

Blaze shows a key management method in an encrypting file system [71]. He states the importance of an effective key escrow mechanism and proposes a smartcard-based key escrow mechanism in a cryptographic file system. Miller et al. [72] show a novel secure network-attached storage system with little penalty to performance. They focus on end-to-end data security and integrity. Kher et al. show a detailed analysis of security on distributed storage systems [73] for governmental or enterprise environments as well as comparisons of each distributed file system.

2.8 Hardware Dependent Problems

Harbitter et al. show performance measurement of public key-enabled Kerberos authentication in a mobile computing environment [74]. They handle PKINIT, which is an extension of the Kerberos protocol [17]. They show the measurement result of the PKINIT implementation on mobile devices. Fox et al. propose a lightweight Kerberos client proxy called Charon for a device with a less powerful processor like a PDA or mobile phone [75]. Standard Kerberized client devices have to interact with KDC (Key Distribution Center), TGS (Ticket Granting Server) and Kerberized service. Therefore, they propose the Kerberized client proxy. A small device like a PDA or mobile phone only needs to interact with the proxy and most processes of Kerberos are done by the proxy.

2.9 Device Management Service

Mei et al. show a remote device management framework [76]. They focus on a management framework like SNMP (Simple Network Management Protocol) [77]. They distinguish entities as manufacturing companies, service providers and customers. The system can access a device's operational status and maintenance function, etc. They employ SyncML (Synchronization Markup Language) [78]. SyncML is a lightweight XML-based data synchronization standard.

2.10 Grid Computing

Forester et al. show a security architecture for computational grids [79] called the Grid Security Infrastructure (GSI). They present an implementation of the architecture within the Globus Metacomputing Toolkit provided by the Globus Alliance. They discuss authentication and access control mechanisms on a grid environment. In grid computing, the entities consist of user, process and resource. Zhao et al. discuss problems with a PKI-based authentication mechanism and proxy certificate approach in grid systems [80].

2.11 Digital Rights Management

Popescu et al. show a security architecture allowing DRM (Digital Rights Management) in home networks [81]. They propose "Authorized Domains." In the authorized domains, the device can transfer the copyrighted data to another device. They propose a hybrid compliance checking and group establishment protocol based on pre-distributed symmetric keys. They also describe the efficient and flexible key updating and revocation method. Abbadi et al. propose a DRM scheme using a mobile device [82]. They define the following entities: a mobile device, the domain, and the domain owner to control domain membership. In this model, the representative mobile device only accesses a centralized server and other devices send a join request to the mobile device. This model also employs physical proximity information. The devices in the same domain can transfer copyrighted data to each other.

2.12 Context-based Approach

Nicholson et al. present transient authentication for mobile device security. They employ a wearable token to verify a user's presence [83]. Jea et al. describe the context-aware access method for public shared devices [84]. A public shared device is shared by the public with many different individuals. In particular, they focus on BSNs (Body Sensor Networks). They show an LED authentication to authenticate between a user and a device. Ficco et al. show a new security device that enables a mobile user to access wireless services using a context-aware approach [85]. They show a prototype implementation using a Bluetooth personal radio identification badge (BRI).

2.13 Others

Tang et al. propose to store a part of a mobile phone's PIN in a remote machine in the network [86]. They state the importance of protecting the PIN because current mobile devices can be used for more important dealings like e-commerce. They show a probabilistic model and the security analysis. Mlakar et al. show a personalized iTV system [87]. They propose a TV device that executes face authentication and automatically collects information on the TV program the user is watching. Suh et al. show the PUFs (Physical Unclonable Functions) for device authentication and secret key generation. The PUFs are circuit primitives that extract secrets from physical characteristics of integrated circuits (ICs). They show the low cost authentication and key-generation method using the PUFs.

2.14 Summary

This dissertation proposes a novel trust model for the inter-device communication paradigm in chapter 5. Chapter 7 discusses the group and domain management method for devices. A federation mechanism such as a discovery and control mechanism is outside the scope of this dissertation, as are network mechanisms like Mobile IP, NEMO and MANET. The dissertation only assumes that each end-point device employs conventional TCP/IP mechanism and does not consider which network mobility mechanism the target system employs. On the other hand, the dissertation assumes that hardware, BIOS, boot loader and OS software on the end-point device are protected and an attacker cannot break into these parts. For example, we can utilize a trusted computing base to protect hardware resources, BIOS, boot loader and OS software. In particular, the combination of an attestation mechanism and TPM is effective to verify integrity for hardware and software components. Because the target devices in this dissertation are assumed to have limited hardware resources, the dissertation discusses problems of hardware performance in chapter 7. Chapter 7 discusses some potential future applications for the inter-device communication paradigm. For example, future distributed networked devices can be used for a grid computing environment. On the other hand, a context-based approach is effective to acquire ownership of the device. Chapter 7 discusses some potential methods for acquiring ownership of the device.

Figure 20 shows the entire security architecture on the inter-device communication and related work. This dissertation focuses on the authentication and authorization mechanism for the inter-device communication paradigm. The proposal also includes network confidentiality and integrity, and an ID protection method on end-point devices based on a tamper-resistant smart card. The dissertation shows demonstration experiments for some home appliances.

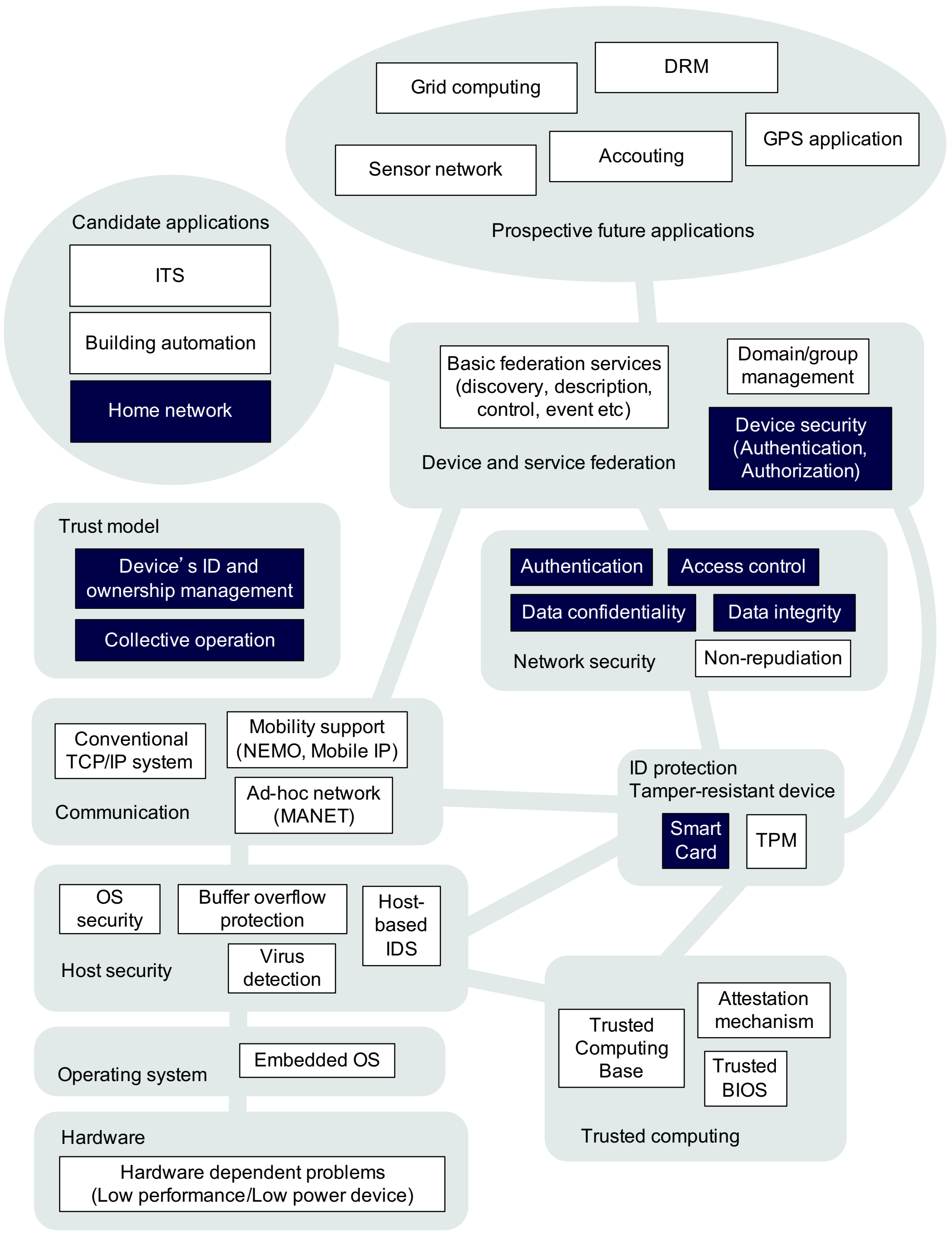


Figure 20. Security architecture and related work for inter-device communication

3. Dissertation's Challenges

This dissertation's main challenge is establishing a novel security mechanism for an inter-device communication paradigm that meets the requirements shown in section 1.5. Figure 21 shows the dissertation's challenges briefly. The challenges consist of two parts. One is the main contribution for novel device-oriented smart card architecture and its user-centric device ID management mechanism. Another is a feasibility study of the overall security system based on the main contribution. This dissertation proposes to employ a conventional network layer's security protocol such as IPsec on the inter-device communication system.

Why does this dissertation employ a network layer's security mechanism such as IPsec? There are many security mechanisms to protect communication channels on the Internet. Figure 22 shows the OSI reference model. A network layer's security mechanism like IPsec [88] can authenticate each *host (device)* entity directly. Therefore, it has suitable characteristics on the granularity of the authenticated entity. The communication between both devices is automatically protected by IPsec-VPN. It stands to reason that the network layer's security is suitable for the inter-device communication. An IP-based security mechanism like IPsec can support many existing networked devices because many networked devices are constructed on a conventional TCP/IP system. On the other hand, the upper layer's security protocol like TLS (Transport Layer Security) [89] can handle *user* entity and *service (server program)* entity directly. However, TLS cannot handle *host (device)* entity directly. Moreover, TLS can only protect a session-based communication channel.

Figure 23 shows a typical host-based security mechanism based on IPsec and IKE. In Figure 23, the authenticated entity is *host* expressed by *Host ID*. All communication channels between both hosts are automatically protected by a host-based VPN service. In this mechanism, each user's datagram is not identified by both hosts. For example, VPN gateway does not identify each datagram on the VPN tunnel. This mechanism is typically used to construct a VPN between geographically distant offices and a remote-access VPN service. This dissertation proposes a novel security mechanism for the inter-device communication in two stages. In the first stage, the dissertation shows an extension method for a conventional network layer's security mechanism. This proposal shows a feasibility

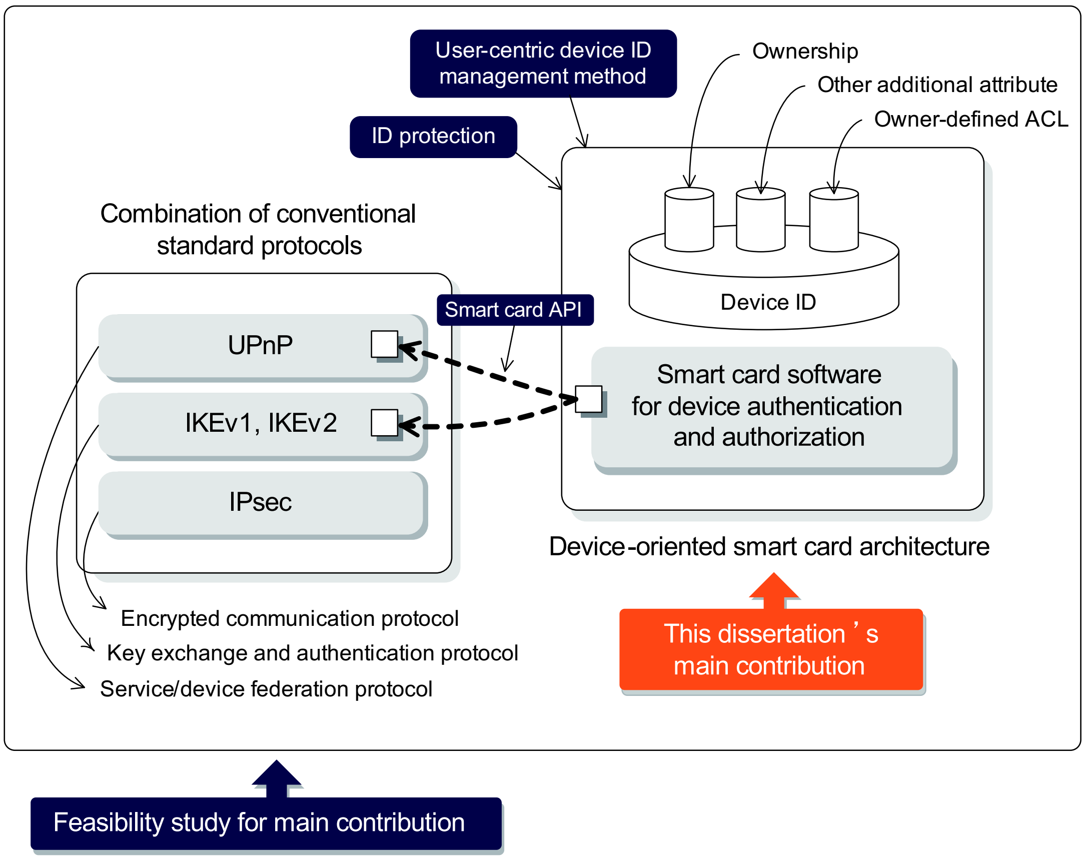

User-centric device ID management method
Ownership
Other additional attribute
Owner-defined ACL
ID protection
Combination of conventional standard protocols
Device ID
Smart card API
UPnP
IKEv1, IKEv2
IPsec
Smart card software for device authentication and authorization
Device-oriented smart card architecture
Encrypted communication protocol
Key exchange and authentication protocol
Service/device federation protocol
This dissertation's main contribution
Feasibility study for main contribution

Figure 21. The dissertation's main contribution and feasibility study

study of a network layer's security mechanism for the inter-device communication system. In the next stage, this dissertation shows a novel security mechanism described in section 1.5. The following sections briefly introduce these challenges.

3.1 Applying Host-based Security Mechanism to User-level Applications

The security mechanism for the inter-device communication basically needs to handle the *host* (i.e. *device*) entity. Moreover, it needs to handle the *service* entity and *user* entity separately. However, a conventional network layer's security mechanism cannot handle *user* and *service (server program)* entity directly as

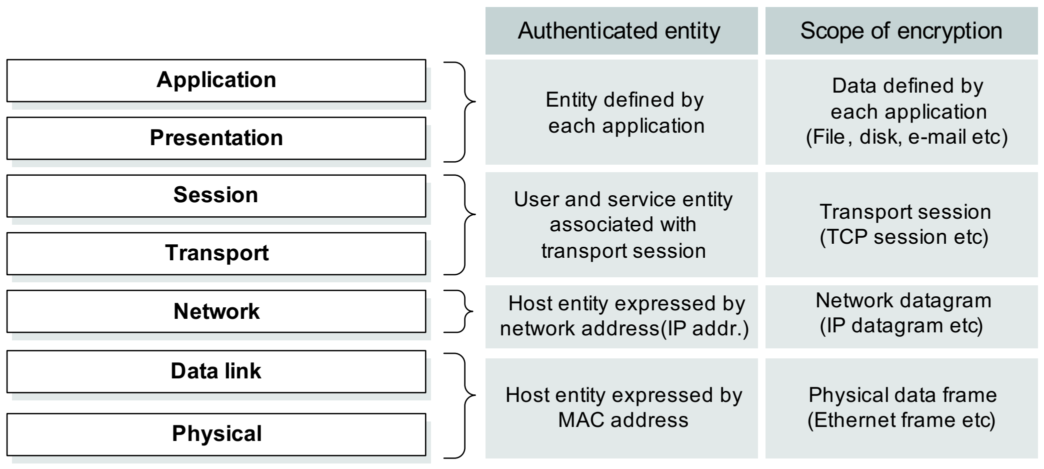

Authenticated entity
Scope of encryption
Application
Presentation
Entity defined by each application
Data defined by each application (File, disk, e-mail etc)
Session
Transport
User and service entity associated with transport session
Transport session (TCP session etc)
Network
Host entity expressed by network address(IP addr.)
Network datagram (IP datagram etc)
Data link
Physical
Host entity expressed by MAC address
Physical data frame (Ethernet frame etc)

Figure 22. OSI reference model and features of security mechanisms

shown in Figure 23. Therefore, this dissertation proposes an extension method for a network layer's security mechanism. The proposal extends a network layer's security mechanism to handle *user* entity and *service* entity for authentication and authorization. Details of the proposal and prototype implementation are shown in chapter 4. Figure 24 shows the extension method for a network layer's security mechanism. In this proposal, each device is controlled by a user directly and each user has a *user ID* on the device. User-to-service authentication and authorization are extension parts of a conventional network layer's security mechanism. An extended security mechanism can handle *device (host)*, *user* and *service* entity directly. Of course, the proposed system can provide a host-based VPN service between both devices. This feasibility study shown in chapter 4 indicates that a network layer's security mechanism has suitable characteristics for the inter-device communication paradigm.

3.2 Novel Inter-device Authentication and Authorization Framework Guaranteeing Explicit Ownership

Section 1.5 shows requirement analysis for the security mechanism on the inter-device communication paradigm. Chapter 5 shows a novel security mechanism

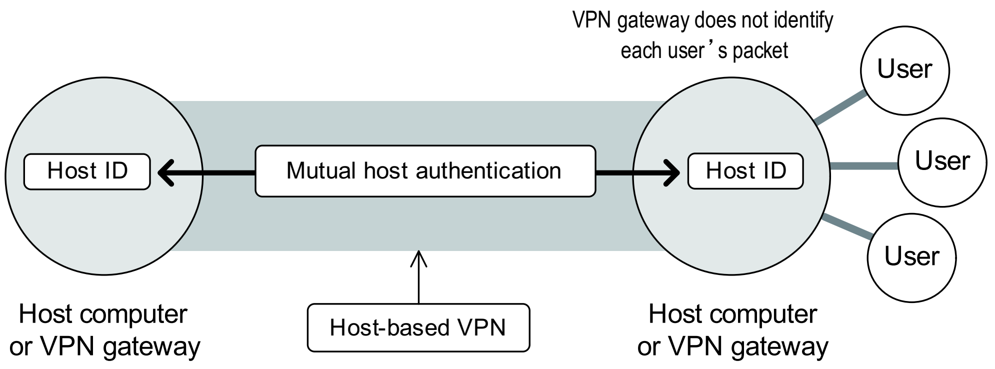

VPN gateway does not identify each user's packet
User
User
User
Host ID
Mutual host authentication
Host ID
Host computer or VPN gateway
Host-based VPN
Host computer or VPN gateway

Figure 23. Typical host-based security mechanism

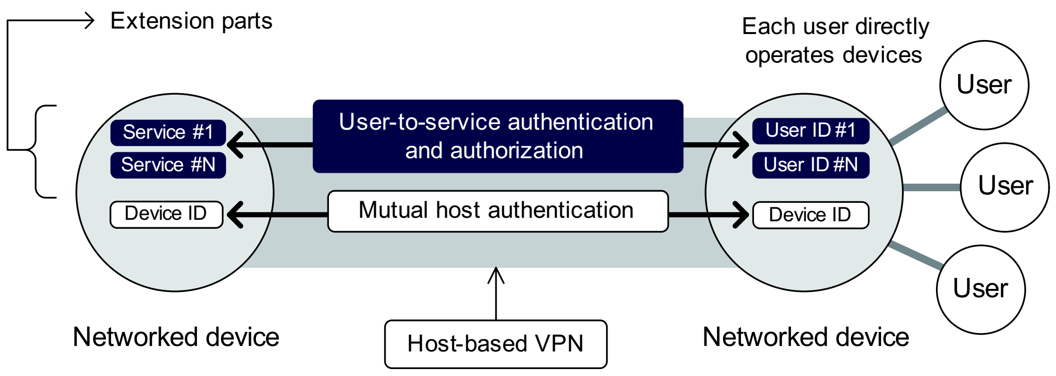


Figure 24. Applying host-based security mechanism to user-level applications

based on these requirements. In the user-controlled devices shown in Figure 24, the user directly utilizes each *user ID* and the service authorizes the access based on the *user ID*. On the other hand, an autonomous device without human interaction requires some kind of delegation mechanism. This dissertation defines a novel *ownership* entity explicitly. The *ownership* guarantees the user's right on the device. In that sense the *ownership* is similar to the *ticket* of Kerberos [17][18] or the *capability* [90]. In the proposal, the user can transfer the user's *ownership* to the *device* and it is associated with the *device ID*. The *ownership* is also associated with the *agent process* requested by the user. Thus, the *agent process* with the *ownership* will be able to act as the user. Figure 25 shows a novel security

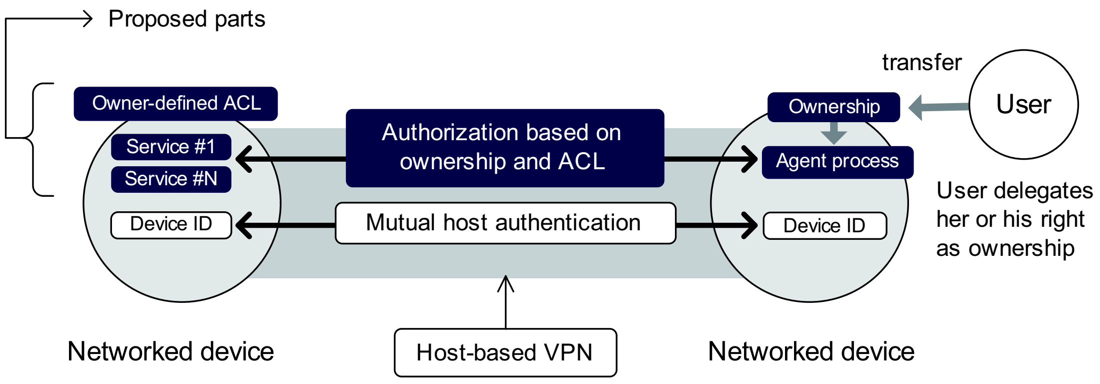


Figure 25. Novel security mechanism for the inter-device communication paradigm

mechanism based on a network layer's security mechanism. The core components of this proposed mechanism aim to guarantee the following security requirements shown in section 1.5: (1) Authentication and authorization in distributed environment, (2) Separation between device's ID and other attributes, (3) Association between device's ID and attributes, (4) ID protection on the distributed device, and (5) Independence from applications and implementations.

Section 1.4 shows some examples of primitive autonomously controlled devices in ITS, BA and home networks. A typical primitive autonomous device has some simple "if-then" rules. If one rule meets the condition, the *agent process* begins to operate its own task with the associated *ownership*. The remote *service* can authorize the access based on the *ownership* of the peer *agent process*. The access control is executed based on an owner-defined ACL (Access Control List) on the device. The details of the proposal and the prototype implementation are shown in chapter 5. The demonstration system of the proposal is also presented in chapter 6.

4. Applying a Host-based Security Mechanism to User-level Applications

4.1 Introduction

IPsec (IP security) [88] and IKEv1 (Internet Key Exchange, version 1) [91] can be used to construct a VPN (Virtual Private Network) [92] service. Although IPsec has transport mode to protect upper layer protocols, it cannot apply a strong authentication and encrypted communication mechanism to user-level applications directly. One reason is that the SA (Security Association) in the IPsec specification targets a host entity only. Therefore, the developer has to implement security mechanisms for each application individually despite the strong security mechanism of IPsec.

This chapter proposes a novel mechanism called the UADB (User Authentication Database). The UADB mechanism extends IPsec and IKE to user-level applications. The UADB mechanism can guarantee a relation between the authenticated user ID and the socket-pair information. Thus, our proposal can provide an access control mechanism based on a user ID for user-level applications. This chapter presents a prototype implementation of the UADB mechanism for WWW client/server programs.

This chapter is organized as follows: Section 4.2 reviews a structure of the IPsec implementation on 4.4 BSD UNIX. Section 4.3 shows the problems with a use of IPsec and IKE for user-level applications. Section 4.4 introduces the UADB mechanism to improve these problems. Section 4.5 shows the design of the UADB mechanism. Section 4.6 presents the prototype implementation. Section 4.7 shows the performance measurement of the prototype. Section 4.8 shows comparison with TLS protocol. Section 4.9 presents an API to apply the UADB mechanism to other existing server applications. Section 4.10 discusses the versatility of the UADB mechanism. Section 4.11 reviews some known problems of the proposal. Section 4.12 summarizes some related work. Section 4.13 concludes the summary.

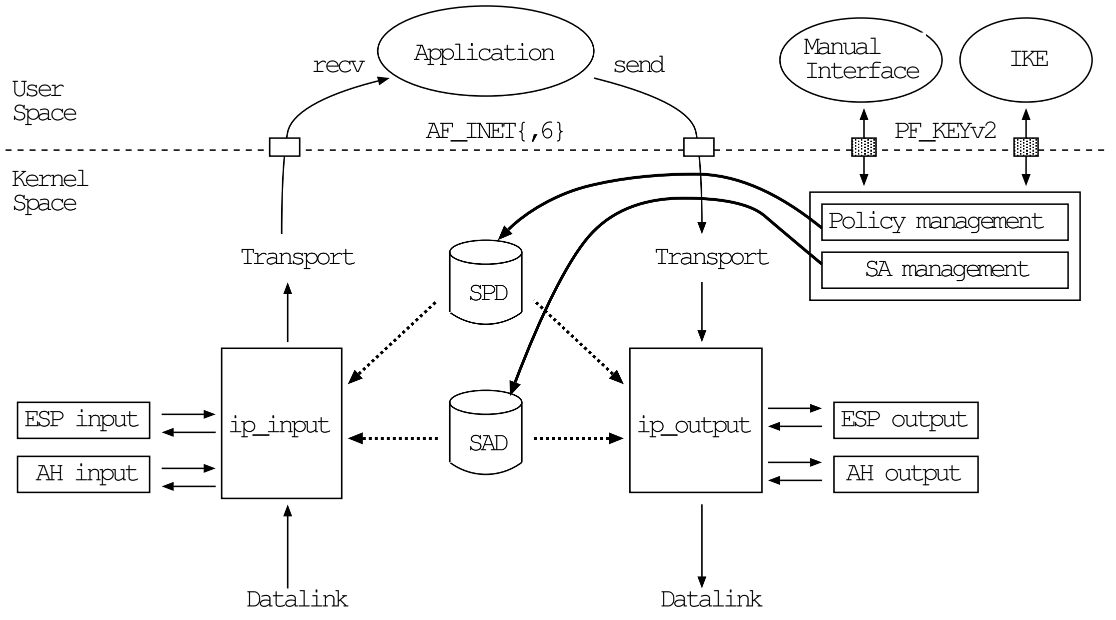


Figure 26. Implementation model of IPsec in 4.4 BSD UNIX

4.2 IPsec Implementation on 4.4 BSD UNIX

IPsec provides a protection method for IP datagrams on an IP layer. IPsec consists of two basic security protocols; AH (Authentication Header) [93] and ESP (Encapsulating Security Payload) [94]. AH provides an integrity check service for IP datagrams. ESP provides a traffic confidentiality mechanism for IP datagrams. IKEv1 (Internet Key Exchange, version 1) protocol processes a mutual host-based authentication and a negotiation of the security parameters such as session keys for AH and ESP. IPv6 (Internet Protocol, version 6) [95] specification includes IPsec as a standard function.

IPsec provides security mechanisms based on an option header mechanism of an IP datagram. This section describes the overview of the KAME [96] IPv6/IPsec implementation. The KAME software was implemented on 4.4 BSD UNIX by WIDE [97] project. Figure 26 shows an implementation model of the IPsec in 4.4 BSD UNIX. AH and ESP of the KAME software is implemented as an advanced feature of the standard TCP/IP function in the 4.4. BSD UNIX kernel.

The kernel with the KAME software has SAD (Security Association Database) and SPD (Security Policy Database). The SAD is used to maintain the SA (Security Association). SA consists of some security parameters like session keys to protect the communication channel between two hosts, encryption algorithm and lifetime of the SA. The SPD is used to maintain security policies for IPsec. The IKEv1 program loads the SPD to decide which packet has to be protected. The IKEv1 program negotiates an appropriate SA with the peer host. The IKEv1 program reads/writes the SAD and the SPD in the kernel by using PF_KEYv2 [98] socket interface. All processes of AH, ESP and IKEv1 are executed based on the rules of the SAD and the SPD.

Here are the processes up until the establishment of the IPsec's protected connection. At first, the IKEv1 program loads the SPD and obtains the security policy. If the packet is matched with the policy, the IKEv1 program begins to negotiate with the peer IKEv1 program. Both IKEv1 programs complete the negotiation and set up the SA onto the SAD in each kernel. Finally, the IPsec software (KAME) begins to apply AH and ESP to the IP datagrams. From the viewpoint of the user-level applications, the IPsec functions are completely transparent. The IP datagrams are automatically encrypted on a network layer.

4.3 Problems

There are two limitations to applying IPsec and IKEv1 to user-level applications. This section describes these fundamental problems.

4.3.1 Problem 1: Authentication for User-level Applications

IKEv1 protocol has 2 phases to establish an IPsec connection. Therefore, the IKEv1 protocol has two kinds of SAs. The IKE-SA consists of security parameters for the phase 1 exchange. The IKE-SA is used to protect the IKEv1 protocol itself. IPsec-SA consists of security parameters for the phase 2 exchange. The IPsec-SA is used to protect the IPsec communication channel.

Figure 27 shows a data-origin authentication mechanism and the host-based authentication mechanism in the IKEv1 protocol. First, an IKEv1 program executes a host-based authentication in phase 1 exchange and establishes the IKE-

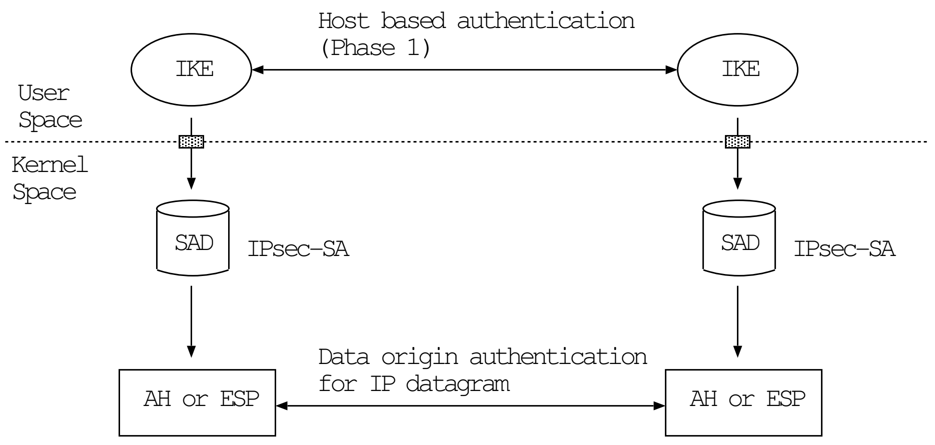

Host based authentication (Phase 1)
IKE
IKE
User Space
Kernel Space
SAD
IPsec-SA
SAD
IPsec-SA
Data origin authentication for IP datagram
AH or ESP
AH or ESP

Figure 27. Data origin authentication in IPsec and host authenticated IPsec-SA by IKEv1

SA. Next, the IKEv1 program establishes IPsec-SA using the encrypted communication channel protected by the IKE-SA. Therefore, AH and ESP process data-origin authentication based on the host-authenticated IPsec-SA. Because the IKEv1 protocol employs only the host-authenticated IPsec-SA, user-level applications cannot employ the authentication mechanism of the IKEv1 protocol directly. In short, user-level applications usually need a user-level authentication function. However, the IKEv1 protocol only provides a host-based authentication function.

4.3.2 Problem 2: Encrypted Communication for User-level Applications

If we apply IPsec to user-level applications directly, a server program cannot identify multiple users from the same host at the same time. This section compares this problem to the TLS (Transport Layer Security) protocol [89]. The TLS protocol processes an encrypted communication channel as a *session* bound to the authenticated user. Figure 28 shows the relation between the session and the authentication mechanism in the TLS protocol. The authentication result between a user and a server program is valid in the *session* of the TLS protocol.

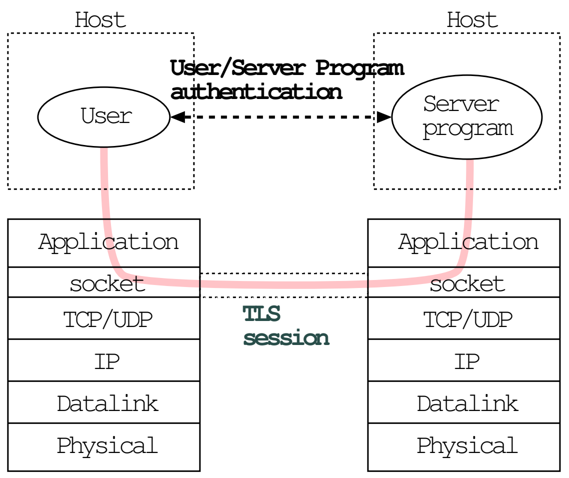


Figure 28. Relation between session and authenticated entities in TLS protocol

On the other hand, IPsec applies its security mechanisms to IP datagrams. Therefore IPsec does not have a *session* mechanism. Figure 29 shows the relation between an IP datagram and the authentication mechanism in the IPsec protocol. The host-based authentication result of the IKEv1 protocol is valid in the IP datagrams. Furthermore, IP header, AH header and ESP header do not have any information for a user ID. Thus, IPsec does not have a mechanism to map a relation between a user ID and an IP datagram. Although IPsec can guarantee the relation between an IPsec-SA and a host, IPsec cannot map an IPsec-SA and a user ID. For these reasons, a server program cannot identify multiple users from the same host at the same time.

4.4 User Authentication Database (UADB): An Extension Method of IPsec and IKEv1 for User-level Applications

Section 4.3 has described that the IPsec and the IKEv1 protocols are host-based security mechanism, therefore, they cannot apply to user-level applications directly. This section presents a novel method to apply host-based security mechanisms like IPsec and IKEv1 to user-level applications. This chapter proposes the

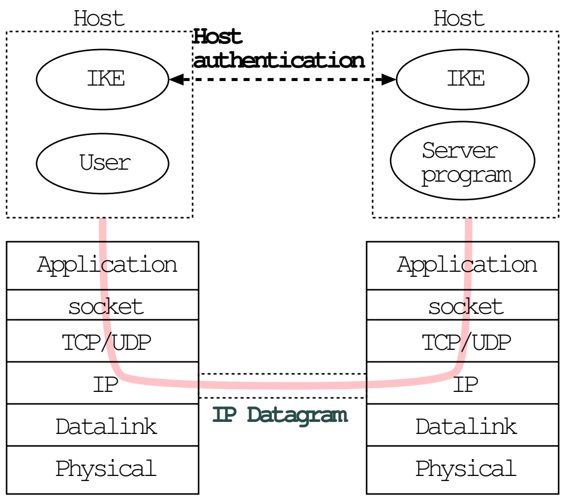


Figure 29. Relation of IP datagram and authentication result in IPsec protocol

UADB (User Authentication Database) mechanism to overcome the constraint described in section 4.3. This chapter especially focuses on the flexibility of the ISAKMP framework [99]. The IKEv1 protocol is basically constructed on the ISAKMP framework. The proposal defines some novel ISAKMP payloads in the IKEv1 protocol to map a user ID and a socket-pair. As a result, we can extend IPsec and IKEv1 to user-level applications without modifying the fundamental framework of the IPsec and the IKEv1 specification.

4.4.1 UADB Database Format

The UADB mechanism treats the parameters shown in Table 1 as a UADB record. The *expiration date* indicates the lifetime of the UADB record. If the UADB record expires, it has to be deleted from the host's UADB. The *User ID in UNIX* or the *DN (Distinguish Name) in X.509 certificate* [19][20] express a user ID. The *Destination IP address* expresses a server's IP address that is connected to by a client. The *Destination port number* expresses the port number that is connected to by a client. The *Source IP address* expresses a client's IP address. *The number of source port numbers* expresses the number of source ports that is stored in the UADB record. The *Source port numbers* expresses the source port numbers

Table 1. Format of UADB record

Expiration date
UserID in UNIX or DN in X.509 certificate
Destination IP address
Destination port numbers
Source IP address
The number of source port number
Source port numbers
Digital signature

that are used by a proxy program shown in section 4.6.1. The proposal utilizes the multiple source-port numbers in a UADB record. Therefore, the UADB mechanism can be able to maintain multiple transport connections virtually, such as TCP or UDP. The *Digital signature* is generated from the hash value of the above information and the sender's private key. The UADB mechanism has two main objectives: (1) A user authentication mechanism based on a digital signature. (2) A mapping mechanism between a user ID and a socket-pair. The following section describes the details of these objectives.

4.4.2 Purpose 1: User Authentication Using Digital Signature

The first objective of the UADB mechanism is to provide a user authentication mechanism based on a digital signature. This mechanism remedies the authentication problem shown in section 4.3.1. Figure 30 shows the user authentication mechanism using a UADB record. The IKEv1 program can authenticate a user in the phase 2 exchange. An IKEv1 program transfers a UADB record to the peer IKEv1 program in the phase 2 exchange. The peer IKEv1 program verifies the digital signature in received the UADB record using the sender's public key. As a result, this step can authenticate the user. If the verification of the digital signature in the UADB record fails (i.e. user authentication fails) then the IKEv1 program discards the UADB record, and they do not establish any IPsec-SA. If the verification is successful then both hosts establish the IPsec-SA

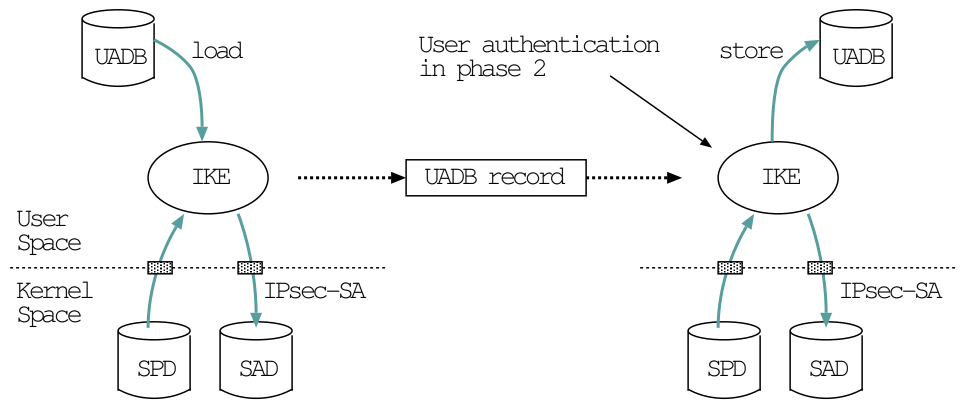


Figure 30. User authentication mechanism using UADB record in IKEv1's phase 2 exchange

and the communication channel is protected by IPsec. In short, the proposal extends the conventional phase 2 exchange in the IKEv1 protocol. The UADB mechanism will be able to establish special IPsec-SA for the authenticated user.

In this step, the IKEv1 programs save the received UADB record onto each host's UADB. The stored UADB record is used for server programs in the next section.

4.4.3 Purpose 2: Mapping Mechanism between User ID and Socket-pair

The second objective of the UADB mechanism is mapping between a user ID and a socket-pair. This mapping mechanism remedies the encrypted communication problem described in section 4.3.2. Figure 31 shows the mapping mechanism between a user ID and the socket-pairs. The socket-pair consists of source IP address, source port, destination IP address, and destination port number. At first, the IKEv1 program on the client host transfers a UADB record to the peer IKEv1 program on a server host. The received UADB record is stored onto each host's UADB. The server program that is connected with the client program searches the socket-pair information in the UADB. If the server program finds an appropriate socket-pair then the server program will be able to map the socket-

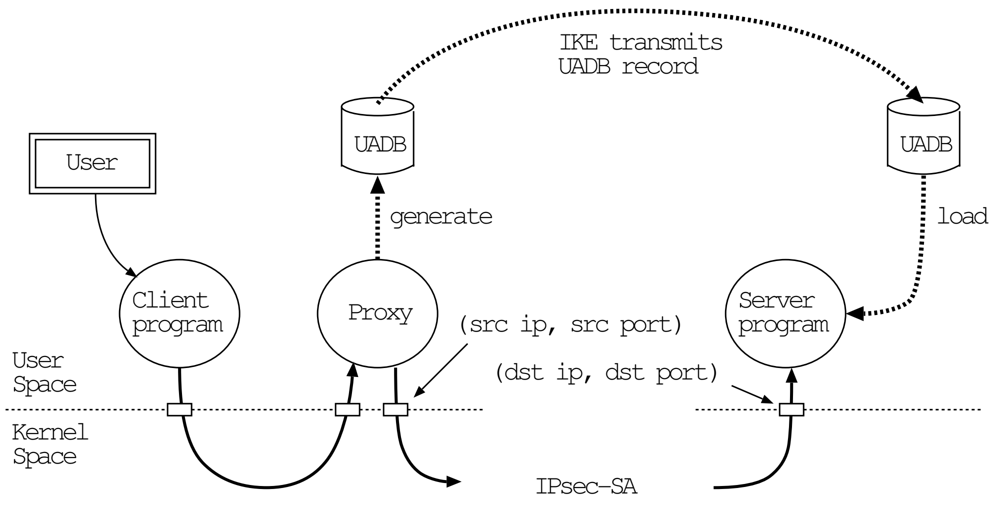


Figure 31. Mapping mechanism between user ID and socket-pair

pair and a user ID. In this step, because the verification of the digital signature in the UADB record is successful, the server program can trust the user ID related with the socket-pair.

4.4.4 Additional ISAKMP Payloads in IKEv1's Phase 2 Exchange

In the proposal, an IKEv1 program transfers a UADB record in the phase 2 exchange in the IKEv1 protocol. Phase 2 is operated as a *quick mode* to negotiate security parameters for an IPsec-SA. The prototype implements additional 2 ISAKMP [99][100] payloads in a first message of the phase 2 (*quick mode*). The additional 2 ISAKMP payloads are used to transfer a user's X.509 certificate and to execute user authentication using a digital signature. Figure 32 shows the modified IKEv1's phase 2 exchange (*quick mode*). The additional 2 ISAKMP payloads are underlined "UADB" and "USER_CERT". The proposal utilizes "UADB" to transfer a UADB record as binary data. The proposal also utilizes "USER_CERT" to transfer a user's X.509 certificate. The IKEv1 protocol negotiates security parameters based on the ISAKMP framework. Therefore, the proposal has to add 2 payloads in IPsec DOI (Domain of Interpretation) [99] to implement the UADB mechanism on the standard IPsec protocol suite.

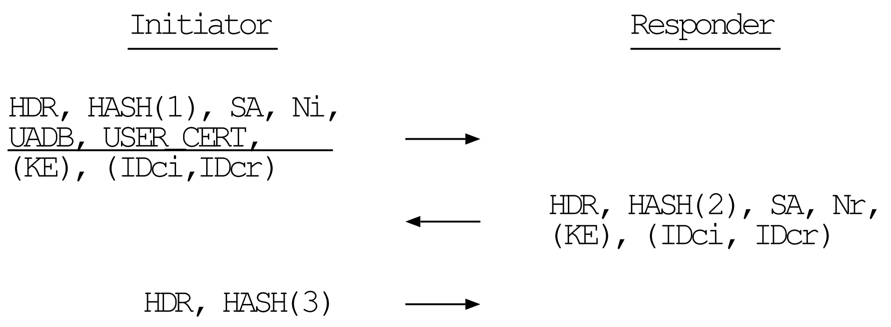


Figure 32. Modified IKEv1 phase 2 exchange (quick mode)

4.5 Design

This section presents the design of the UADB mechanism described in section 4.4. The proposal applies IPsec and IKEv1 for an encrypted communication channel and a user authentication to user-level applications. The section applies the UADB mechanism to WWW applications and ensures that the prototype implementation can work on an IPv6 network.

4.5.1 Initial Configuration

A user has to prepare the user's private key, the user's X.509 certificate and the configuration file in the user's home directory. In the configuration file, a user has to specify IP addresses that she or he would like to connect. The IP address can be expressed in an IPv4 or an IPv6 format. The administrators of both hosts have to set up security policies for the IPsec protocol using the IPsec policy management program. A trust anchor CA (Certificate Authority) certificate also has to be stored on a server host.

4.5.2 Process Flow on Client Host

Figure 33 shows the process of a client host. The client's process is executed according to the following procedure.

1. A web browser tries to connect to an HTTP daemon on the server via a

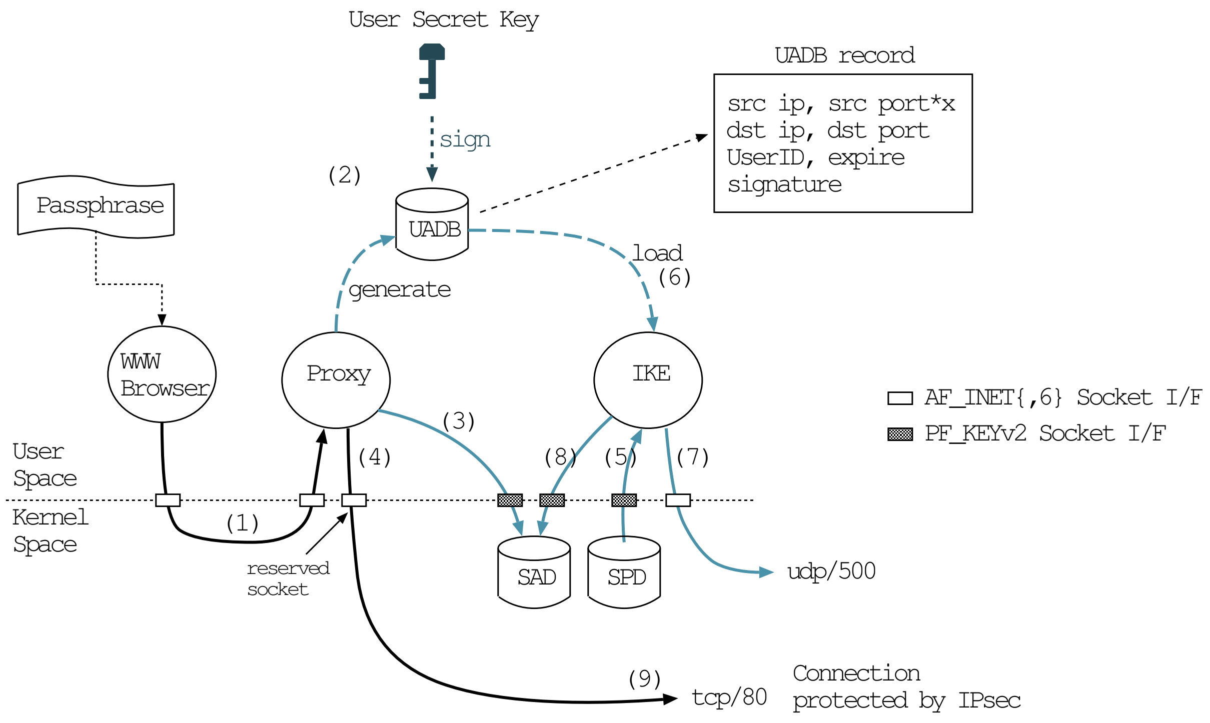


Figure 33. Process flow on client host

proxy.

2. The proxy searches for the user ID, the destination IP address and the destination port number as search keywords in the host's UADB. If there is no appropriate UADB record then it begins to generate a new UADB record. At first, a proxy displays an authentication web page to request a passphrase for the user. The proxy loads the user's secret key and decrypts by the passphrase. Next, the proxy generates reserved ports described in section 4.6.1. At last, the proxy makes a UADB record and attaches a digital signature to the last part of it. The proxy saves the created UADB record onto the host's UADB.

3. The proxy sends an "SADB_DELETE" message using the PF_KEYv2 socket interface [98] to delete the existing IPsec-SA of the same socket-pairs. The PF_KEYv2 is a special socket interface to maintain an SAD and an SPD in a kernel.

4. The proxy connects to the HTTP daemon using the reserved sockets. At this time, the request is blocked because no IPsec-SA is established yet.

5. The kernel issues an "SADB_AQUIRE" message to the IKEv1 program to request a new IPsec-SA.

6. The IKEv1 program loads the appropriate UADB record from the host's UADB.

7. The IKEv1 program begins to negotiate a new IPsec-SA with the peer host. First, the IKEv1 program executes mutual host authentication in the phase 1 exchange. Next, the IKEv1 program transfers the UADB record and the user's X.509 certificate using the additional 2 ISAKMP payloads.

8. The IKE program issues "SADB_ADD" to the kernel and stores a new IPsec-SA onto the kernel's SAD.

9. The blocked connection between the web browser and the HTTP daemon is established. This connection is encrypted and integrity-checked by IPsec.

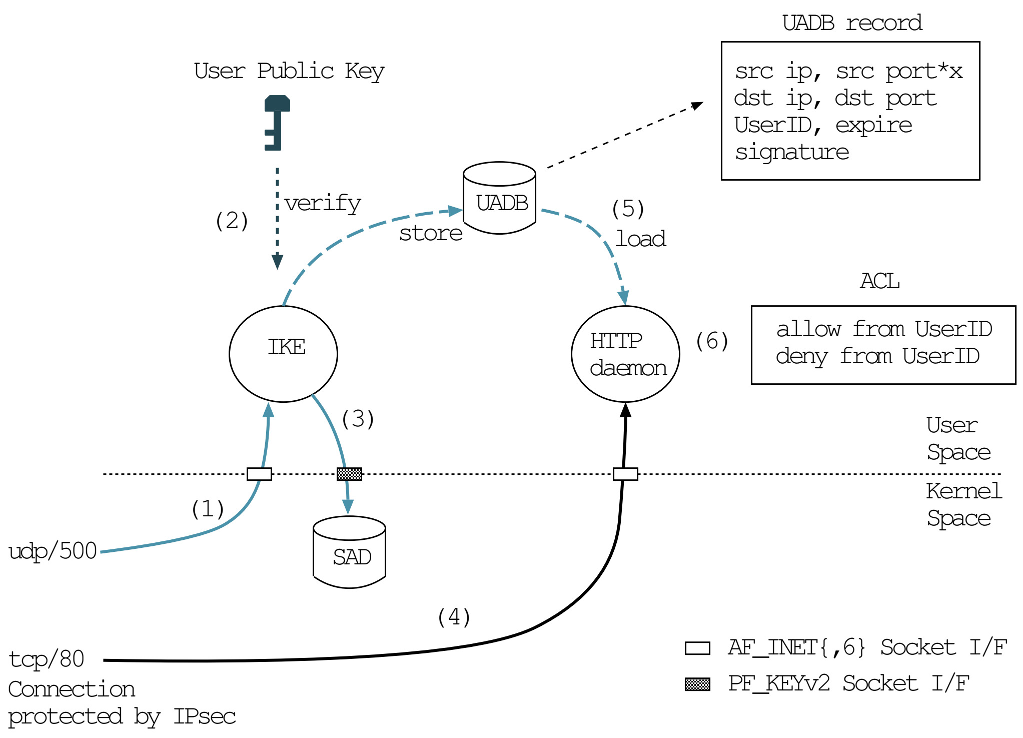


Figure 34. Process flow on server host

10. The client host deletes an expired UADB record periodically. The client host also deletes a UADB record that depletes the reserved-sockets.

4.5.3 Process Flow on Server Host

Figure 34 shows the process flow on a server host. The server's process is executed according to the following procedure.

1. An IKEv1 program on a server host receives an initial message from a connecting client host.

2. The IKEv1 program executes mutual host authentication and establishes the IKE-SA in the phase 1 exchange. Next, the IKEv1 program receives a UADB record and the user's X.509 certificate in the phase 2 exchange.

```
order deny,allow
deny from all
allow from {bob,alice}
```

Figure 35. Access control list based on UADB-extended user ID. (".htaccess" file)

The IKEv1 program verifies the user's X.509 certificate using the trust anchor CA certificate. Then, the IKEv1 program verifies the digital signature to authenticate the user using the user's public key. If the user authentication succeeds, then the IKEv1 program checks the expiration date of the UADB record and the limitation number of reserved source ports. If there is no problem, the UADB record is stored on the host's UADB. If the user authentication fails, then the IKEv1 program stops the negotiation immediately.

3. The IKEv1 program establishes the IPsec-SA. The IKEv1 program issues "SADB_ADD" to the kernel and stores the IPsec-SA on the host's SAD.

4. After establishing the IPsec-SA, the connection from the client program to the server program is established. This connection is encrypted and integrity-checked by IPsec.

5. The HTTP daemon checks the connection with the client and obtains the socket-pair information. Then, the HTTP daemon searches the socket-pair in the host's UADB. As a result, the HTTP daemon will be able to identify the user ID from the socket-pair information using the found UADB record.

6. The HTTP daemon loads the ACL (Access Control List). Then, the HTTP daemon executes an access control based on the URL and the user ID. Figure 4.5.3 presents an example usage of the ACL.

7. The server host deletes an expired UADB record periodically. The server host also deletes a UADB record that depletes the reserved-sockets.

Table 2. Software used for the prototype implementation

OS	FreeBSD 4.3 Release
IPv6/IPsec stack	KAME
IKEv1 program	racoon-20011026a
Proxy program	tinyproxy-1.3.3b
HTTP daemon (server program)	apache_1.3.20 + IPv6 patch
Web browser	Mozilla 0.8.1(Gecko/20010419)
Cryptographic library	OpenSSL 0.9.6

4.6 Prototype Implementation

To show the usability of the UADB mechanism, this section implements the UADB mechanism for a WWW service. The proposal mainly implements the following three parts: (1) A proxy program that works as a wrapper for a client program (i.e. a web browser). The proxy program requests the passphrase for the user and generates the UADB record. (2) Some modifications of the IKEv1 program. The proposal adds the transfer function for newly-defined ISAKMP payloads and the user authentication function on the IKEv1 program. (3) Some extended parts of a server program (i.e. a HTTP daemon). The proposal includes an access control module based on a URL and a user ID.

In summary, the proposal needs the proxy program, modified IKEv1 program and a few modifications of the server program. However, our proposal does not need any modification of the kernel software. Table 2 shows the software used for the prototype implementation.

4.6.1 Source Port Problem and Proxy-based Solution

Section 4.4.1 has described that a UADB record has multiple source port numbers. An IKEv1 program transfers a UADB record to a server host to identify a user ID from the socket-pair information. This process has to be done **before** the client program connects to the server program. Usually, the client program makes a *connect* system call, then the operating system selects a source port number from the unused ports automatically. Therefore, it is difficult to transfer a UADB record that includes the reserved source port numbers.

For this reason, this section proposes a novel proxy mechanism for the reserved socket management. Figure 36 shows the process flow. First, the client program connects the proxy. The proxy searches for the user ID, the destination IP address and the destination port number as a search keyword in the host's UADB. If appropriate UADB record does not exist then the proxy begins to generate a new UADB record. The proxy generates the specified number of sockets by consecutive *socket* system calls, and the sockets are bound to randomly selected source port numbers using *bind* system call. The proxy program stores the sockets (socket descriptors) on the memory for future use. The proxy generates a new UADB record based on the reserved socket information and stores it onto the host's UADB. Next, the IKEv1 program transfers the UADB record to the server host, and the IPsec-SA is established. After the establishment of the IPsec-SA, the client program connects to the server program via the proxy. The proxy connects the server program using the reserved sockets per the HTTP connection. As mentioned above, the server program obtains a socket-pair and it is bound to the user ID. Thus the IKEv1 program will be able to transfer the UADB record that includes the source port numbers before the client actually connects the server program.

4.6.2 Implementation on Client Host

This section has implemented the following parts of the proxy program: the request function for user passphrase, the UADB record generation function, the UADB input/output function, the socket management function and the SAD operation function. Figure 37 shows the authentication page. This section has also implemented the following parts of the IKEv1 program: the loading function of a UADB record from the host's UADB and the UADB transfer function using the extended ISAKMP payloads.

4.6.3 Implementation on Server Host

This section newly implemented the following parts in the IKEv1 program: the transfer mechanism for the UADB record and the user's X.509 certificate, the

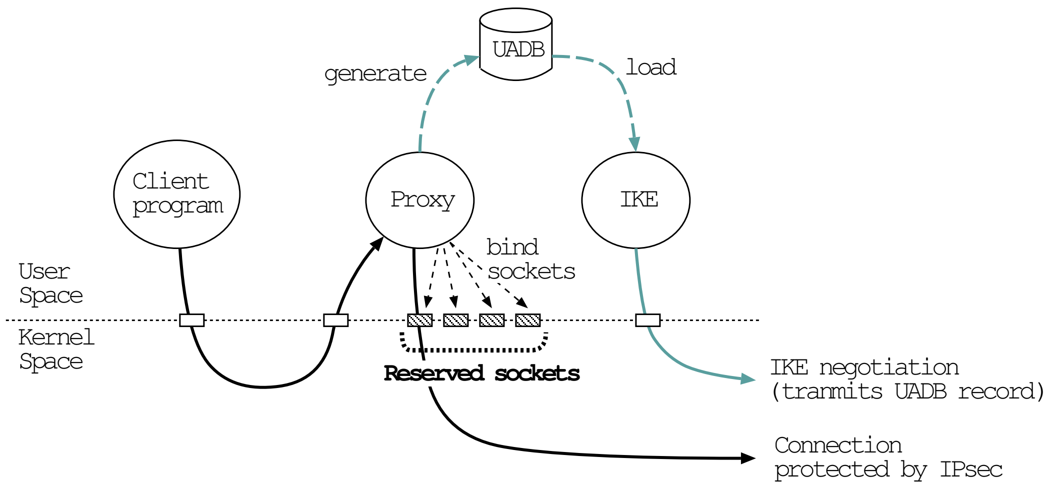
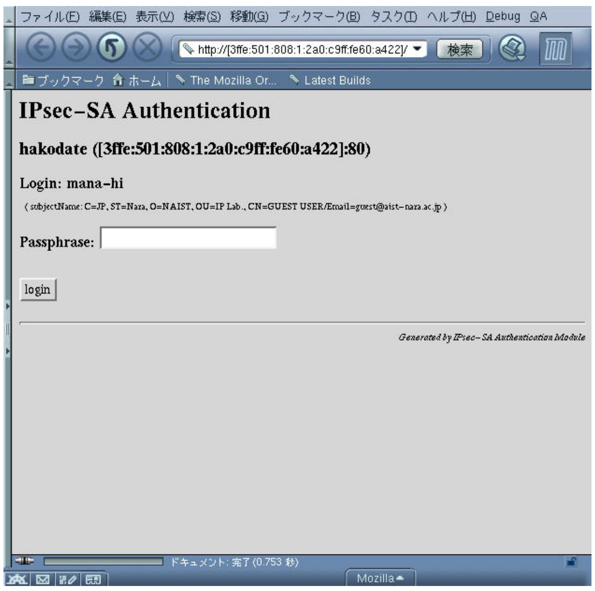


Figure 36. Proxy mechanism for the reserved socket management


ファイル(F) 編集(E) 表示(V) 検索(S) 移動(G) ブックマーク(B) タスク(T) ヘルプ(H) Debug QA
http://[3ffe:501:808:1:2a0:c9ff:fe60:a422]/
検索
ブックマーク ホーム The Mozilla Or... Latest Builds

IPsec−SA Authentication

hakodate ([3ffe:501:808:1:2a0:c9ff:fe60:a422]:80)

Login: mana−hi

(subjectName: C=JP, ST=Nara, O=NAIST, OU=IP Lab., CN=GUEST USER/Email=guest@aist−nara.ac.jp)

Passphrase:

Generated by IPsec− SA Authentication Module

ドキュメント: 完了 (0.753 秒)
Mozilla

Figure 37. Authentication page displayed by the proxy program

Table 3. Hardware specification

	Clinet host	Server host
CPU	Pentium II 333 MHz	Pentium 233 MHz
Memory	256MB DRAM	96MB DRAM
Network	100BaseT	100BaseT

verification function for the user's X.509 certificate and the digital signature in the UADB record, the validation function of the expiration date in the UADB record, the check functions for the limitation number of source port and the input/output function for the UADB. This section also implemented the following parts in the HTTP daemon: the search function for the user ID by the socket-pair information, and the access control module based on the user ID.

4.7 Performance Measurement

This section presents the result of the performance measurement. Table 3 shows the hardware specification used in the measurement. Two computers are deployed on the same segment of the experimental IPv6 network. The prototype implementation employs a 1024 bit RSA public key algorithm for the digital signature.

4.7.1 The Result

Table 4 shows the processing times of a proxy program. The generating process of a UADB record consists of the following processes: the loading process of the user's private key, the decrypting process of the protected private key, the generating process of the digital signature and the storing process of the UADB record. On the other hand, the searching process of the reserved socket consists of the loading process of the UADB records, the searching process of the reserved socket and the updating process of the host's UADB.

Table 5 shows the processing times of the phase 2 exchange of the IKEv1 program. Here are additional functions on the client side: the loading process of the UADB record and the user's X.509 certificate, and the generating process

Table 4. Processing time of customized proxy program

Process	Time [ms]
Generating process of a UADB record	28
Searching process of the reserved socket	19

Table 5. Processing time of phase 2 exchange in IKEv1 protocol

Modification	Time [ms]
Before	356
After	676

of newly-defined ISAKMP payloads. Here are additional functions on the server side: the verification process of the received user's X.509 ceritificate, the verification process of the digital signature in the received UADB record and the storing process of the UADB record.

Table 6 shows the processing time to search for the user ID from the socket-pair information. This function consists of the following processes: the checking process of the socket-pair information using *getpeername* and *getsockname* system calls in the HTTP daemon, the loading process of the UADB record, the checking process of expiration date in the UADB record, the searching process of the user ID and the updating process of the host's UADB.

4.7.2 Comparison with Conventional System

In a conventional system, the IKEv1 program executes mutual host-based authentication in the phase 1 exchange and establishes the IPsec-SA in the phase 2 exchange. The modifications of the conventional system are: (1) The generating process of the UADB record and the searching process of the reserved sockets in the proxy program. (2) The transfer process of the UADB record and the verification process of the digital signature in the IKEv1 program. (3) The map-

Table 6. Processing time to search user ID from socket-pair

Process	Time [ms]
Searching process of user ID	120

ping process between the user ID and the socket-pair information in the HTTP daemon. This section shows the extra processing time that the proposal needs.

First, this section compares the processing time to establish IPsec-SA with a conventional system. The proposed system needs an extra 367 ms. It consists of the generating time of the UADB record in the IKEv1 program (28 ms), the searching time of the reserved socket in the proxy program (19 ms) and the extra time of the phase 2 exchange in the IKEv1 program (320 ms).

Next, this section compares the processing time after the establishment of the IPsec-SA. The proposal added the mapping process of the user ID and the socket-pair information in the HTTP daemon. The HTTP daemon needs an extra 120 ms for every request that makes an *accept* system call.

The total processing time from the establishment of the IPsec-SA to the first HTTP connection establishment is 487 ms. Once the IPsac-SA is established, the HTTP daemon needs only 120 ms for every accepted connection.

4.8 Comparison with TLS Protocol

To utilize the conventional IKEv1 program, the administrators in both hosts have to set up the host's X.509 certificate and the host's private key. To utilize the UADB mechanism, in addition, the user has to set up her or his X.509 certificate and the private key. On the other hand, to utilize the TLS protocol, the administrator has to set up the host's X.509 certificate and the host's private key in a server host. If the sever program needs a user authentication then the user has to set up her or his X.509 certificate and the private key in a client host.

The UADB mechanism achieves user authorization by guaranteeing the mapping between the socket-pairs in the transport layer and the user ID. The IKE program also executes mutual host-based authentication. The TLS protocol achieves the client-to-server authentication by the handshake protocol implemented in the upper part of the transport layer.

A main difference between the proposal and the TLS protocol is the authenticated entity. The proposal has a two-step authentication function. The phase 1 exchange executes the host-to-host authentication in the network layer. The phase 2 exchange executes the user-to-host authentication by the UADB mechanism. The TLS protocol has only one-step authentication, and it can execute the client-to-host authentication directly. However, in most cases, the TLS protocol only executes the unilateral server authentication.

4.9 API Proposal

This section has presented the prototype implementation of the UADB mechanism for a WWW service. The proposal can integrate applications other than the WWW service. However, to integrate the authentication function of the UADB mechanism, we need to modify the server program. Therefore, this section proposes the API for the server applications. The API makes it easier to integrate the proposal. Figure 38 shows the UADB API. The developer has to implement the following parts to integrate the proposal to the application: (1) The extension of the proxy program. The proxy needs to process a target protocol other than the WWW service. (2) The extension of the sever program. The server program has to be integrated into the UADB functions using the proposed UADB API. The server program with the UADB API will be able to identify the user ID from the socket-pair information. The server program can utilize the user ID for access control for its own service.

4.10 Versatility of the UADB mechanism

The proposed UADB is a simple mechanism that provides a mapping method between a socket-pair and a user ID. Therefore, we can deploy the UADB mechanism to applications other than the WWW service. We just have to add the UADB API shown in Table 38 after making the *accept* system call. The prototype was implemented on the FreeBSD operating system. The prototype implementation employs the KAME software for IPv6/IPsec functions and *racoon* for the

SYNOPSIS

char∗ **get_uadb_user_id**(*int* sock,*int* idtype)

DESCRIPTION

This function searches for the host's UADB from the sock parameter, returns user ID, and deletes the used source port number from the UADB and updates the host's UADB. The id-type parameter decides which type of the user ID (The Unix's user ID or the X.509 certificate's DN) is used.

RETURN

The user ID string. If the sock (the socket-pair) is not authenticated then this function just returns -1.

Figure 38. UADB API for server programs

IKEv1 program. As described in section 4.6, we do not need any modifications for the kernel and the IPv6/IPsec software. The proposal has modified the IKEv1 protocol based on the generic ISAKMP framework. Therefore, we can apply the UADB mechanism to another IKEv1 implementation on another platform effectively.

4.11 Discussion

Because the UADB mechanism maintains the critical information in the host's UADB, we have to consider the safer management method of the UADB. The prototype implementation maintains the UADB as a file. Therefore, if an attacker obtains the administrator privilege on the target host then he or she can access the UADB directly. To prevent the attack, this section proposes the kernel-embedded UADB. This section also proposes an access method for the kernel-embedded UADB to extend the conventional PF_KEYv2 interface. If the proposal implements the UADB in the kernel then it will be more difficult to attack. Because other process working with administrator privilege cannot access the UADB directly, the attacker needs to issue the special PF_KEYv2 system call. Moreover, a typical program without the IKEv1 program usually does not includes the PF_KEYv2 system call in the code. Therefore, it will be more difficult to attack the UADB.

On the other hand, the UADB mechanism has a scalability problem in the reserved socket management. This problem has its root in the limitation of the number of sockets that the operating system can maintain at the same time. One simple solution is to resolve the limitation of the number of sockets in the kernel. In most non-proprietary operating systems like Linux or BSD, the developer can rebuild the kernel and improve this problem directly.

4.12 Related Work

There are some mechanisms to authenticate a client entity by a server entity using an independent security protocol from the application protocol. Kerberos [17] is a major independent security mechanism for many applications. Microsoft Windows 2000 and the later versions have the same kind of security mechanism. The difference between these systems and the proposal is the mapping method for the user ID. Although the proposal employs the socket-pair information in the transport layer, Kerberos employs the application-level resource identifier. The advantage of the proposal is versatility. The developer only needs to integrate the UADB API in the target server programs. However, the proposal does not apply the access control function for more abstracted resources rather than socket-pairs such as a file.

4.13 Summary

This chapter has presented a novel framework to integrate the encrypted communication and the authentication function of the IPsec and the IKEv1 protocols to user-level applications. To employ the UADB mechanism, application programs do not have to implement its own security mechanism for encrypted communication and the user-based and host-based authentication mechanisms. This chapter has also presented the prototype implementation of the UADB mechanism for WWW service. Thus, we will be able to apply the strong security mechanism in the network layer to many existing applications by using the UADB mechanism based on the IPsec and the IKEv1 protocols. This chapter has also proposed the UADB API to integrate the proposal to server applications other than the WWW service.

5. Inter-device Authentication and Authorization Framework Guaranteeing Explicit Ownership

5.1 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 a device's identity and an owner's identity clearly because most devices have a single ownership or multiple ownerships. Therefore, this dissertation 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 chapter presents a new concept for an inter-device authentication and authorization framework.

Figure 39 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 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 chapter presents a new attempt to bind the device's identity and its ownership information efficiently. This chapter proposes a novel smart card software

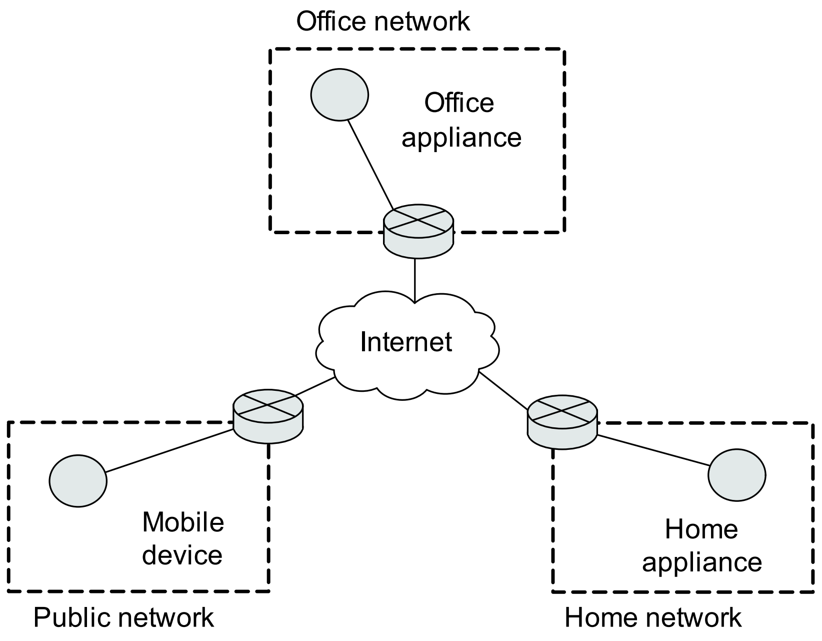


Figure 39. Concept of an inter-device authentication framework

to realize inter-device authentication based on the device's identity, and authorization based on its binding. This chapter also shows how to apply our smart card to the existing standard security protocols.

The rest of this chapter is organized as follows: Section 5.2 shows some practical device authentication mechanisms and a brief idea of multiple ownerships model for inter-device communication. Section 5.5 describes some hardware-based security mechanisms with PKI support and explains why the proposal employs a smart card. Section 5.6 reviews the recent related work on device authentication. Section 5.7 introduces the inter-device authentication framework and its applications. Section 5.9 shows the design of our framework. This chapter presents the prototype implementation of the novel smart card software and its configuration tools, and an authentication middleware system in section 5.10. Section 5.11 shows the result of performance measurements. Section 5.12 discusses technical issues. Section 5.13 concludes with a summary.

5.2 Practical Model for Existing Device Authentication Mechanisms

This section first reviews device authentication mechanisms in some major existing home network specifications. This section then compares these specifications with a practical model for device authentication.

UPnP Security [38] UPnP (Universal Plug and Play) Security specification has a sophisticated mechanism to guarantee 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 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 the 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 [14] 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 and Anonymous) with configurable access rules.

Bluetooth [101] 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.

5.3 Comparison Details

In many existing authentication systems for the client-server paradigm, a user is simply authenticated by a server. However, this dissertation discusses a security mechanism for an inter-device communication, for example, a communication between a home robot and a home server. This section shows the typical procedure of an inter-device authentication and the access control in Figure 40. At first, an owner takes ownership of the device (establishing a secure relationship between the device and the 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 7 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.

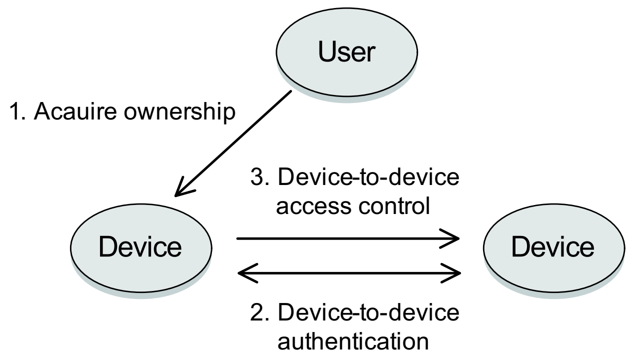

User
1. Acauire ownership
3. Device-to-device access control
Device
Device
2. Device-to-device authentication

Figure 40. Typical device authentication and access control model

Table 7. Comparison of device authentication mechanisms

	UPnP	ECHONET	Bluetooth
Ownership acquisition	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	-	-	-

5.4 Multiple Ownerships Model for Inter-device Communication

Figure 41 shows an inter-device authentication/authorization model from the perspective of the relationship between a device and the owner's identities. A device is mostly owned by someone. Therefore, the device's behavior is logically related, not with a *device identity* but with an *owner 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 identity*. The rest of this section compares the single ownership model with the multiple ownerships model shown in Figure 41.

At first, (A) in Figure 41 shows the single ownership model. Some existing authentication/authorization mechanisms described in section 5.3 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 access control based on a peer *device identity.* Next, (B) in Figure 41 shows the multiple ownerships model the dissertation 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 the proposal adopts 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's ownership. This chapter shows more practical example of our proposal in section 5.8. This chapter proposes 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 identity and its multiple ownerships.

5.5 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

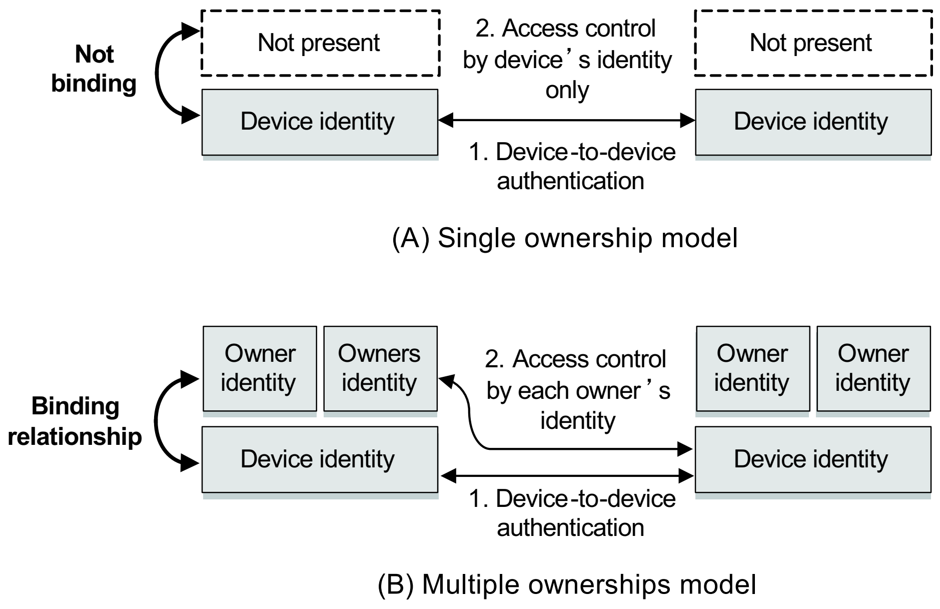

Not binding
Not present
2. Access control by device's identity only
Not present
Device identity
Device identity
1. Device-to-device authentication
(A) Single ownership model
Binding relationship
Owner identity
Owners identity
2. Access control by each owner's identity
Owner identity
Owner identity
Device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

Figure 41. Inter-device authentication/authorization model

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 [**102**] 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 [**103**] Sun Microsystems's Java Card plat-

Table 8. Comparison of hardware-based security mechanisms

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

form technology enables 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 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 [70] The TPM is defined as a microcontroller that stores keys, passwords and digital certificates. The TPM can store the user's or 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 8 shows a comparison of the hardware-based security mechanisms described above. Java Card is a convenient choice to the write code and prototyping for a secure protected execution environment. Thus the proposal employs the Java Card platform technology to develop novel smart card software which realizes our novel inter-device authentication framework, described in section 5.4. Although the TPM's platform attestation is an essential component in every modern device's security, the platform attestation is outside the scope of the dissertation.

5.6 Related Work

Related research on device authentication/authorization is available. Mayrhofer [104] 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 [105] 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 2 devices to improve security rather than one shared key for all devices. Fuji Xerox [106] 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. [107] 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 PKI technology.

5.7 Overview of an Inter-Device Authentication Framework Guaranteeing Explicit Ownership

This section is a brief summary of a proposed inter-device authentication framework. In the proposal, a production-level device's identity is guaranteed by a X.509 Public Key Certificate [19] [20] generated by a manufacturer. The device's personal ownership is guaranteed by Attribute Certificates [108]. 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. The proposal employs Attribute Certificate as key technology to distinguish and bind the device's identity and the ownership information securely. Hereafter, the following section abbreviates the X.509 Public Key Certificate as PKC, the Attribute Certificate as AC.

Figure 42 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 the ACL. Although the proposal 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, the prototype implementation guarantees the pairing between the smart card and the device by the smart card's PIN (owner's PIN) code.

Figure 43 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. The proposal assumes that the product is shipped with this initialized smart card. After purchasing

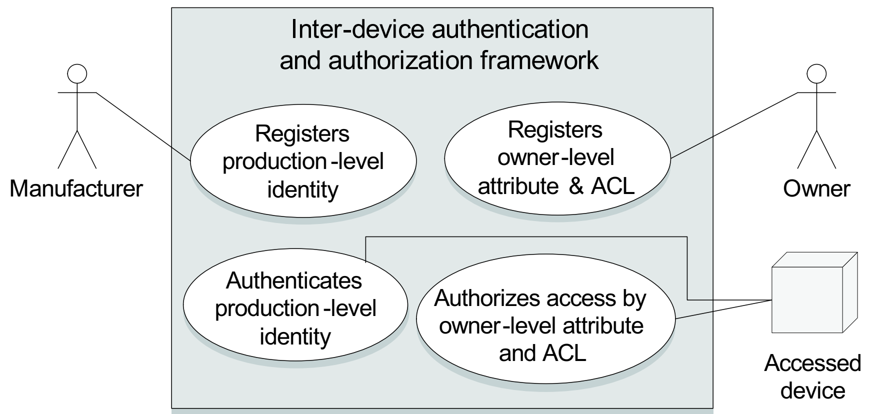

Inter-device authentication and authorization framework
Manufacturer
Registers production-level identity
Registers owner-level attribute & ACL
Owner
Authenticates production-level identity
Authorizes access by owner-level attribute and ACL
Accessed device

Figure 42. Use case diagram for the proposed system

the product, in the ownership acquisition phase, an owner registers her or his 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 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 operational phase, the owner attaches the smart card to the purchased appliance. The 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. The following sections describe the details of the operational phase.

5.7.1 Production-level Identity Authentication using PKC

Figure 44 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

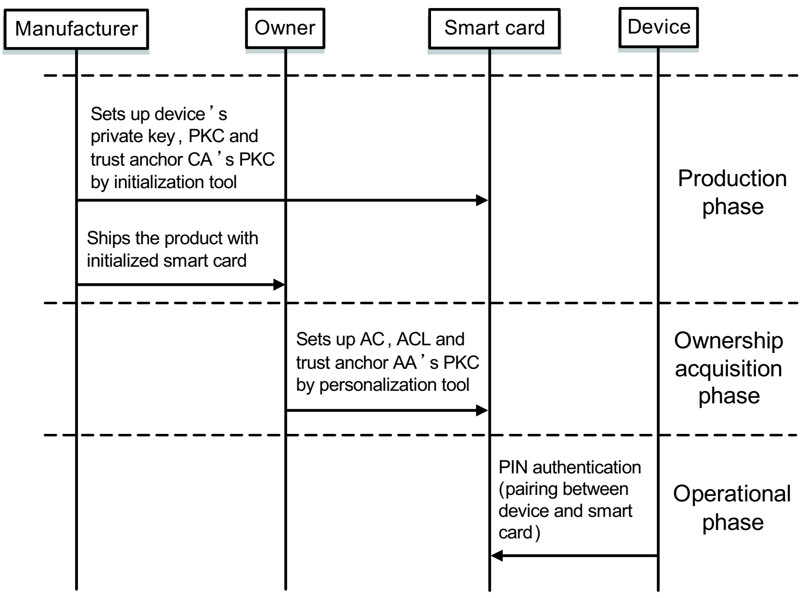

Manufacturer
Owner
Smart card
Device
Sets up device's private key, PKC and trust anchor CA's PKC by initialization tool
Ships the product with initialized smart card
Production phase
Sets up AC, ACL and trust anchor AA's PKC by personalization tool
Ownership acquisition phase
PIN authentication (pairing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

Figure 43. Sequence diagram for the proposed system

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. Section 5.9.3 shows the design of the device authentication middleware system.

5.7.2 Owner-level Attribute Authorization using AC

Figure 45 shows an authorization process based on 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

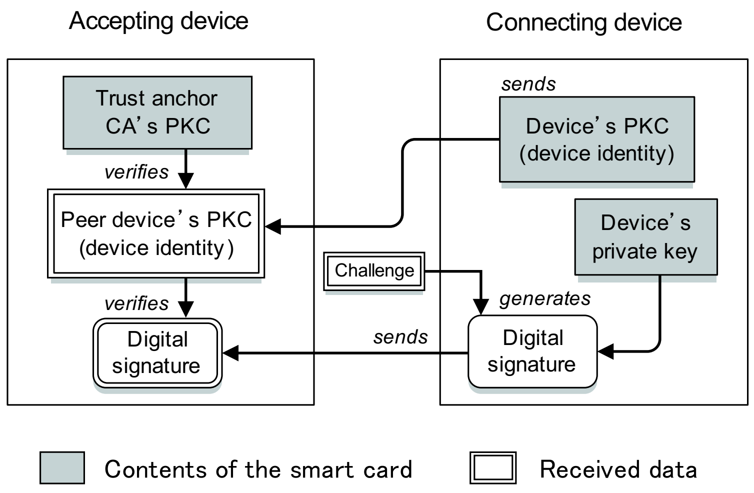


Figure 44. Production-level identity authentication using PKC

the application. For multiple ownerships on a single device, this dissertation proposes that the AC (ownership) which is sent from the connecting device be automatically selected by a program based on a pre-configured rule. The rule maintains the mapping between each operation type and its ownership.

5.8 Example Usage of the Proposal

Figure 46 shows an example usage of the 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 Figure 45. The security company's video server and the resident's TV set are the *accepting devices* in Figure 45. This section considers the following two situations: First, the camera periodically sends pictures to the security company's video server for recording. 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

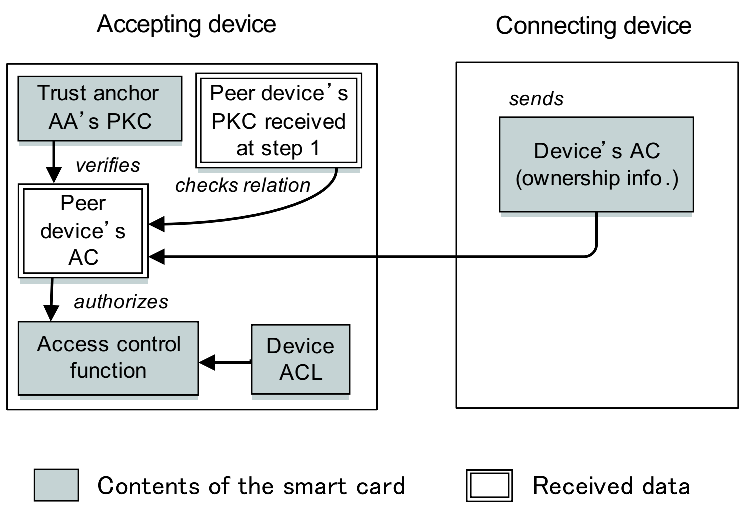


Figure 45. Owner-level attribute authorization using AC

with the security camera, and it is bound to the camera's device identity (PKC).

Figure 47 shows ACL examples in the proposal. The ACL#1 shows ACL entries on the security company's video server. When the camera sends pictures to the company's server to record pictures, 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 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.

5.9 Design

Proposed framework described in the section 5.7 consists of novel smart card software for the device's security, some configuration tools and an authentication middleware system for the smart card. This section shows the design of each component.

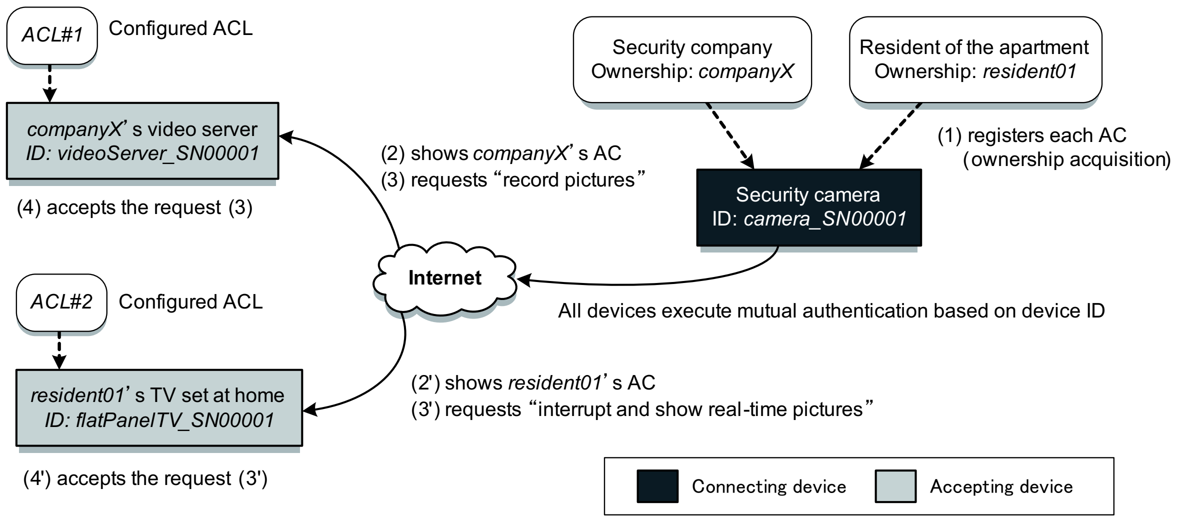

ACL#1
Configured ACL
companyX' s video server
ID: videoServer_SN00001
(4) accepts the request (3)
ACL#2
Configured ACL
resident01' s TV set at home
ID: flatPanelTV_SN00001
(4') accepts the request (3')
(2) shows companyX' s AC
(3) requests "record pictures"
Internet
(2') shows resident01' s AC
(3') requests "interrupt and show real-time pictures"
Security company
Ownership: companyX
Resident of the apartment
Ownership: resident01
(1) registers each AC (ownership acquisition)
Security camera
ID: camera_SN00001
All devices execute mutual authentication based on device ID
Connecting device
Accepting device

Figure 46. Example usage

5.9.1 Novel Smart Card Software for Devices' Security

The proposal employs a smart card with tamper-resistant characteristics to execute secure authentication and access control functions. Figure 48 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 a 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 [109] standard interface via a smart card reader.

Table 9 shows the authentication module API defined in the proposed smart card software. Table 10 shows the authorization module API, and Table 11

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";

Figure 47. Example usage of the ACL syntax

Table 9. 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
storeTrustAnchorCACert	store trust anchor CA's PKC
retrieveTrustAnchorCACert	retrieve trust anchor CA's PKC
getSignature	generate signature and retrieve it

shows the security policy module API. The proposal defines two PIN types in our smart card software. The initialization tool executes *generateKeyPair*, *retrievePublicKey*, *storeDeviceCert*, *storeTrustAnchorCACert* in Table 9. 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. The proposal assumes that the *owner PIN* is shared by the device owners, therefore, the *owner PIN* can restrict unauthorized access to the smart card.

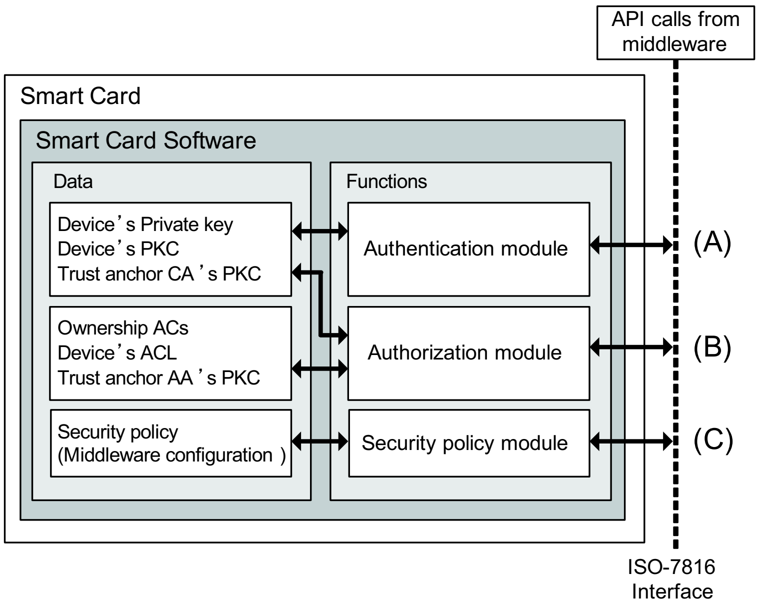


Figure 48. Design of novel smart card software for device security

5.9.2 Configuration Tools for the Smart Card

This section proposes two configuration tools for the smart card described in section 5.7.

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

Table 10. Authorization module API provided by our smart card software

API	Description
storeOwnershipCert	store AC
retrieveOwnershipCert	retrieve AC
storeTrustAnchorAACert	store trust anchor AA's PKC
retrieveTrustAnchorAACert	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 11. Security policy module API provided by our smart card software

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

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 the AC. The owner inputs a unique nickname as an attribute string in the AC to take her or his 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.

5.9.3 Inter-Device Authentication Middleware System

This section proposes an inter-device authentication middleware system using original smart card software. Proposed authentication middleware system is executed on a Linux box (e.g. a micro server). The proposal assumes 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. The prototype implementation employs the proven IKEv1 (Internet Key Exchange, version 1) [91] protocol to execute a mutual authentication between the two devices. The IKEv1 is used as an automatic key exchange to establish an the IPsec [88] secure connection. The proposal also employs 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 5.7.1.

Figure 49 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 Figure 48 and Figure 49) 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) [98] 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. The proposal modifies only the authentication process of phase 1.

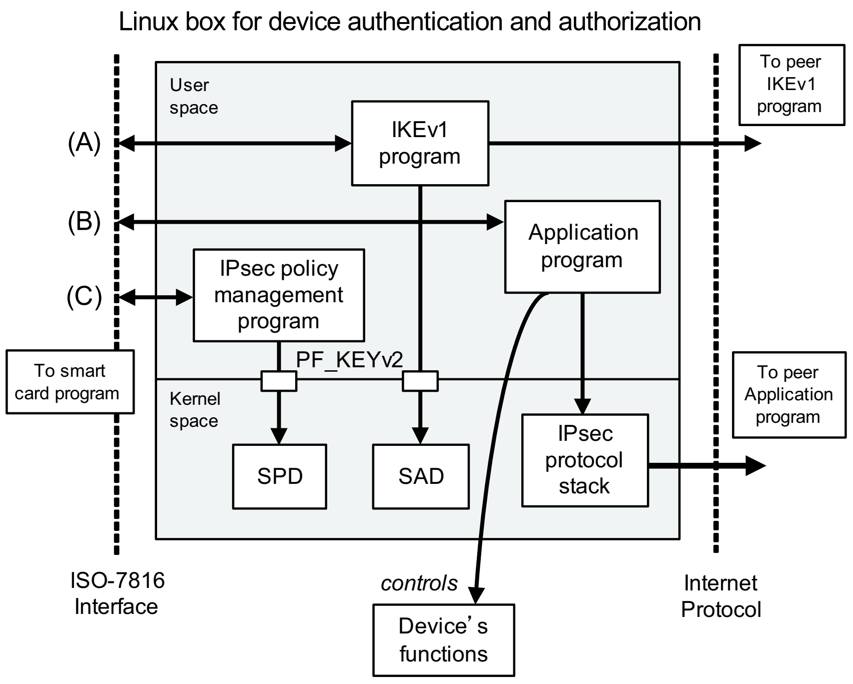


Figure 49. Design of the inter-device authentication middleware system

During the negotiation in phase 1, the IKEv1 program calls the authentication module from the smart card ((A) in Figure 48 and Figure 49). The proposal employs “main mode“ as IKEv1's phase 1 exchange [91]. Figure 50 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) [98] in the Linux kernel. Finally, the communi-

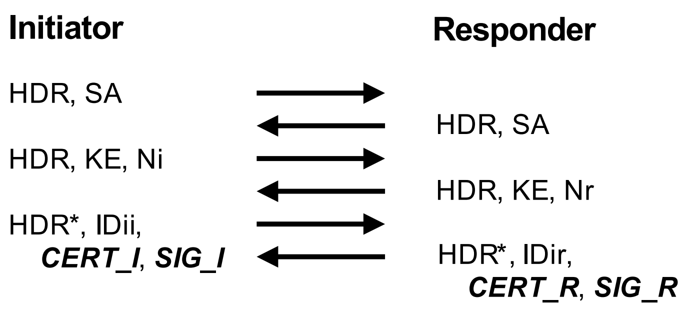


Figure 50. Main mode with signature authentication on IKEv1 protocol

cation channel between the two devices will be protected by IPsec.

5.10 Implementation

This section shows the prototype implementation of the proposal.

5.10.1 Novel Smart Card Software for Device Security

This section shows the prototype implementation of the smart card software described in section 5.9.1. It works on a Gemalto Cyberflex Access e-gate 32k smart card and an Gemalto e-gate token connector as a smart card reader. The proposed smart card software employs the Java Card API 2.1.1. The proposal utilizes the smart card as a small USB device, as shown in Figure 51. The Java applet running on the smart card includes the following functions: 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.

5.10.2 Configuration Tools for the Smart Card

This section shows the prototype implementation of the initialization tool and the personalization tool. They are Java programs running on a Linux operating system. Figure 52 shows the initialization tool's GUI. Firstly, the manufacturer inserts a smart card with an initial state into a reader, and inputs an *administrator*

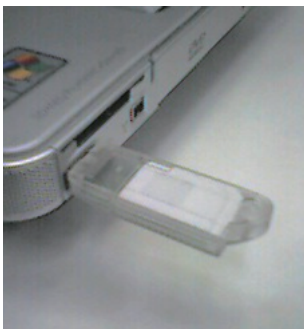


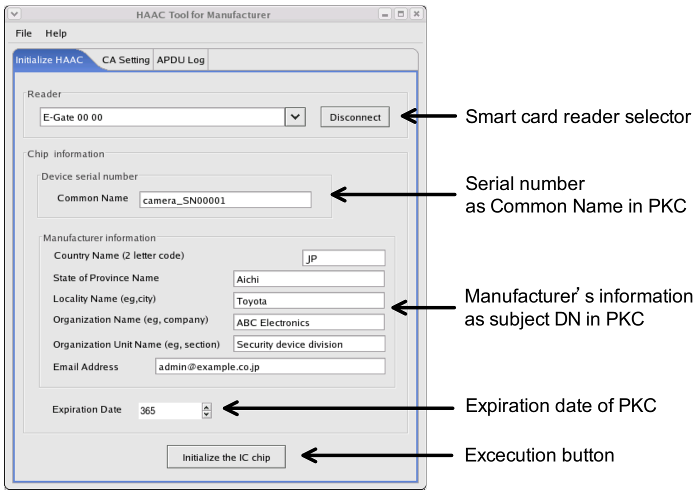
Figure 51. Smart card as a USB device

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 53 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 her or his attribute (nickname for ownership information), and can store the attribute as an AC. The user can also input ACL entries in text format 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.

5.10.3 Inter-Device Authentication Middleware System using IKEv1 and IPsec

Table 12 shows the software used with the middleware implementation described in section 5.9.3. *USAGI* [110] is for the implementation of an IPv6 protocol stack with an IPsec function on the Linux platform. The prototype implementation employs the *USAGI*'s IPsec to secure the communication between the devices.



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

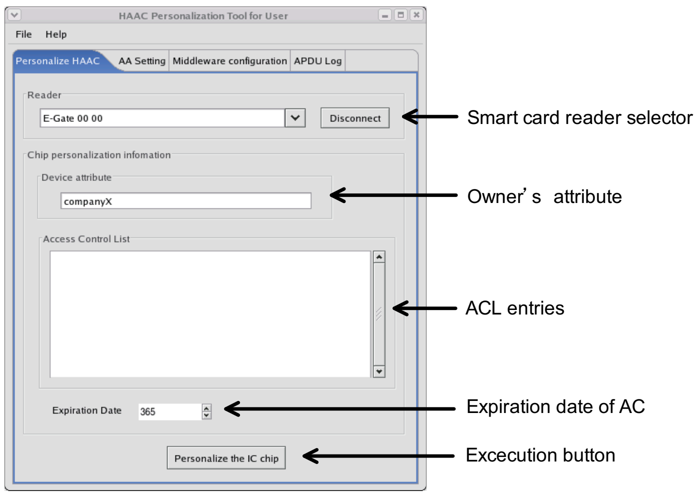
Expiration Date: 365

Figure 52. Initialization tool for the smart card

Racoon [111] is the proven and major implementation of the IKEv1 protocol. The proposal currently employs the IKEv1 protocol. To employ the new IKEv2 protocol [112], the developer has 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 1024-bit RSA-SHA1 signature/verification algorithm supported by the smart card. The prototype implementation utilizes *setkey* [111] 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. The prototype implementation employs the *PCSC-Lite* library [113] to utilize the smart card reader on the PC platform.

5.11 Performance Measurement

This section shows the results of the performance measurement of the authentication middleware system shown in section 5.10.3. Table 13 shows the hardware



Reader: E-Gate 00 00

Device attribute: companyX

Expiration Date: 365

Figure 53. Personalization tool for the smart card

specification of the Linux boxes. The measurement utilizes two Linux boxes with a configured smart card and deployed the middleware. Table 14 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 Figure 44. 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 Figure 44. The measurement employs a 1024-bit RSA-SHA1 signature algorithm. Table 15 shows a comparison between the proposed system and the conventional system in total time of mutual device authentication. This result expresses the total process time to establish a connection in phase 1 of IKE while using the main mode with signature authentication as shown in Figure 50. 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 Figure 50) 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 Figure 50) at the IKE program initiation. Therefore, the result of the proposed IKEv1 program does not include

Table 12. Software used with the middleware implementation

Platform	Linux (kernel 2.6.11)
IPv6/IPsec protocol 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
CCID driver	ifd-egate-0.05

Table 13. Hardware specification of the Linux boxes

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

the loading time of the device's PKC.

5.12 Discussion

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

5.12.1 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 current implementation, this dissertation proposes 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

Table 14. Process time of the basic smart card functions

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

Table 15. Comparison of the process time of IKE phase 1 exchange

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

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 to 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.

5.12.2 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 5.2 guarantee only the single ownership of a device. The proposed framework can distinguish clearly between the device's identity and its ownership information by using the PKI technology. In the proposal, the ownership information (AC) is strongly associated with device identity (PKC). It is 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, this chapter has proposed a framework that enables the verification of the ownerships of a device. This chapter has also shown a prototype implementation for an inter-device authentication middleware system using devices' identities and an access control method based on the ownership information.

This chapter employs a smart card to achieve original device security functions based on the PKI technology. To employ the PKI technology, the proposal is somewhat complex, so this dissertation has developed some configuration tools to reduce the user's burden in managing the smart card. An owner poses 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 has the problem that it requires substantial manual inputs from the manufacturer. The proposal needs to integrate automated registration procedures for high-volume manufacturing. In addition, the current personalization tool can execute on a local machine only. To make the proposal more convenient, the system needs to support a remote update mechanism for a device's ACL on a smart card.

5.12.3 PKI Operational Issues

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

Multiple Manufacturer's CA The proposal guarantees the device's production-level identity by using a PKC that is issued by the manufacturer's CA. If multiple manufacturers deploy proposed framework, then it must consider 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 [114]. The proposal also has 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).

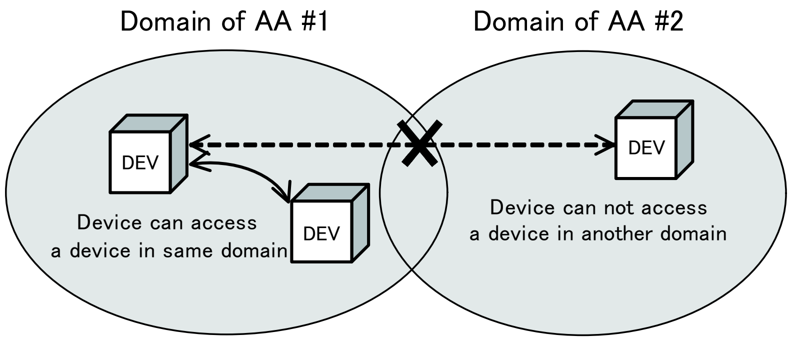


Figure 54. Multiple owner's AA

Multiple Owner's AA The 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 cannot 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 54 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.

5.13 Summary

This chapter has analyzed the existing device authentication mechanisms. A single device has its device identity and is normally bound to a single owner identity or to multiple owners' identities. This chapter has 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.

This chapter has 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. The proposal provides production-level the device's identity authentication using the PKC, and an access control mechanism based on the ownership information using AC. The ownership information (AC) is strongly associated with the device's identity (PKC). It 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 ownership information and device identity.

To realize the proposal, for the prototype implementation this chapter has developed 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 chapter 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. This is the first step towards the utilization of the framework. It is a fundamental mechanism for inter-device authentication and authorization which will contribute to the development of secure Internet-ready appliances and their middleware systems. Next chapter presents more practical demonstration systems.

6. Demonstration Systems of the Inter-device Authentication and Authorization Framework

6.1 Introduction

Chapter 5 has presented the Inter-device authentication and authorization framework that supports distinguishing between, and binding, the device's identity and the ownership information. This chapter shows some applications for actual home appliances and presents the results of the demonstration experiments. Chapter 5 has shown the prototype implementation using the IKEv1 program. At first, this chapter extends the prototype to the latest version of the IKEv2 (Internet Key Exchange, version 2) protocol [112] in section 6.2. Next, section 6.3 presents the design of the demonstration system. The dissertation targets two kinds of appliances: conventional home appliances without network functions and the Internet-ready appliances. Section 6.4 gives the details of the implementation for the demonstration experiments. Section 6.5 presents the results of the demonstration experiments. Section 6.6 concludes the chapter.

6.2 Handling the IKEv2 Protocol

Chapter 5 has presented the prototype implementation of the Inter-device authentication and authorization framework using the IKEv1 program. This section shows the prototype implementation using the latest version of the IKEv2 protocol. Version 1 of the IKE protocol was defined in RFC 2407, RFC 2408 and RFC 2409 [99][100][91]. The conventional IKEv1 protocol has 2 phases and consists of some sub-protocols, so it has complex architecture. Therefore, the conventional IKEv1 protocol has the problem that it is difficult to validate the security by mathematical method [115]. Processes of the IKEv2 protocol is more simplified than conventional IKEv1 protocol. The IKEv2 protocol has only 1 phase to establish CHILD-SA (formerly known as the IPsec-SA in the IKEv1 protocol).

Figure 55 shows the detail of the IKEv2 protocol. The IKEv2 protocol executes mutual authentication and the establishment of the IKE-SA in the first and second exchanges. The IKEv2 protocol executes the establishment of the

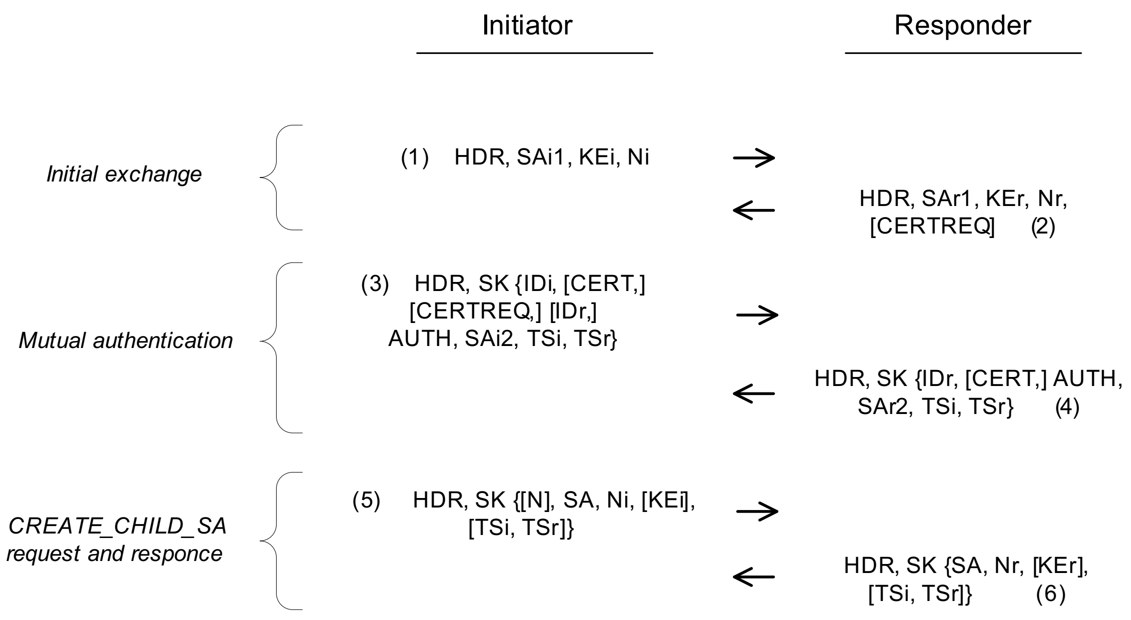

Initiator
Responder
Initial exchange
(1) HDR, SAi1, KEi, Ni
HDR, SAr1, KEr, Nr, [CERTREQ] (2)
Mutual authentication
(3) HDR, SK {IDi, [CERT,] [CERTREQ,] [IDr,] AUTH, SAi2, TSi, TSr}
HDR, SK {IDr, [CERT,] AUTH, SAr2, TSi, TSr} (4)
CREATE_CHILD_SA request and responce
(5) HDR, SK {[N], SA, Ni, [KEi], [TSi, TSr]}
HDR, SK {SA, Nr, [KEr], [TSi, TSr]} (6)

Figure 55. The IKEv2 protocol

CHILD-SA in the third exchange.

This section extends the following parts of the IKEv2 protocol. We do not modify any standard procedures in the IKEv2 protocol.

- At the start-up time of the IKEv2 program, it loads the device's PKC and the trust anchor CA's PKC from the smart card.
- The IKEv2 program (the initiator and the responder) sends the device's PKC as a CERT payload in the message (3) or (4).
- The IKEv2 program (the initiator and the responder) calculates the digital signature using the authentication module API in each smart card. This is an AUTH payload in the message (3) or (4).
- The peer IKEv2 program (the initiator and the responder) receives the message (3) or (4). The IKEv2 program saves the received peer device's PKC onto the local disk.

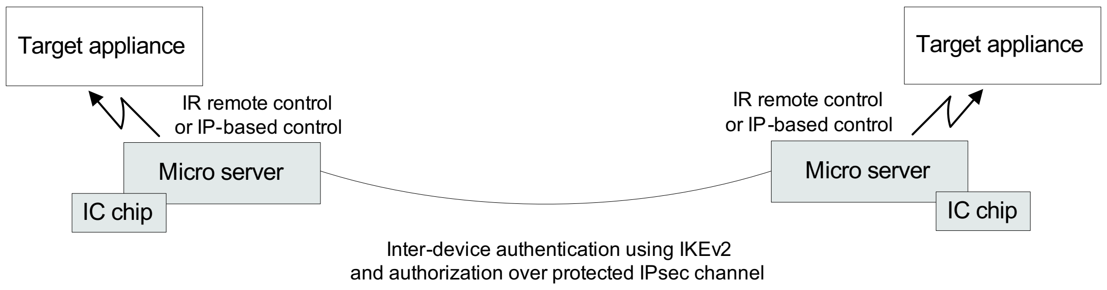

Target appliance
Target appliance
IR remote control or IP-based control
IR remote control or IP-based control
Micro server
Micro server
IC chip
IC chip
Inter-device authentication using IKEv2 and authorization over protected IPsec channel

Figure 56. Basic components of the demonstration systems

- The IKEv2 program (the initiator and the responder) first verifies the received CERT payload in the message (3) or (4) (i.e. the peer device's PKC) using the trust anchor CA's PKC. If the verification is successful then the IKEv2 program verifies the AUTH payload in the message (3) or (4) using the peer device's public key. If the verification is successful, then inter-device authentication is completed.

6.3 Design of Demonstration Systems

This section shows the design of the demonstration systems for the inter-device authentication and authorization framework.

6.3.1 Basic Architecture

Figure 56 shows the basic components of the demonstration systems. The demonstration system employs a micro server to apply the framework to conventional appliances. The micro server is a small Linux box and it also has an IC chip (smart card). The IC chip has the target device's PKC, the trust anchor CA's PKC, the ACs (the ownership information) and the trust anchor AA's PKC. The middleware system shown in section 5.9.3 is installed onto the micro server. The demonstration system this chapter shows employs the IKEv2 program. Thus, the micro server can execute inter-device authentication using the IKEv2 protocol and the device's PKC. The micro server can control the target appliances by the IR remote control function or the direct IP-based control function. This

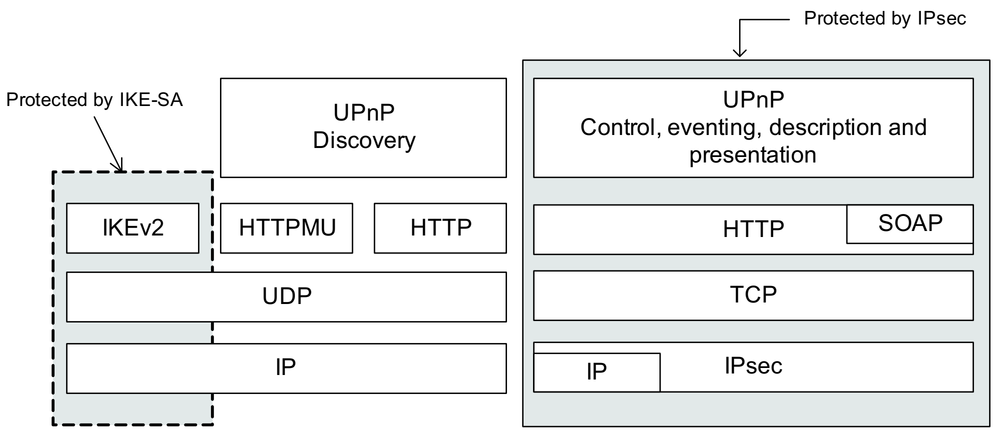


Figure 57. Inter-device authentication and authorization framework and UPnP framework

chapter presents the design of both methods. The rest of this chapter especially describes in particular the detail of the authorization mechanism.

6.3.2 Demonstration System for Non-networked Conventional Home Appliances

This section first shows the design of the demonstration system for non-networked conventional home appliances. The demonstration system employs the IR remote control to control the target appliance. The demonstration system also employs the UPnP framework to control the appliance. We can utilize any framework to control actual appliances other than UPnP. However, to show the versatility of the framework, this chapter employs the UPnP protocol as a typical standard home network protocol.

Figure 57 shows the relation between the inter-device authentication and authorization framework and the UPnP framework. At first, the IKEv2 program executes the inter-device authentication and establishes the CHILD-SA on both micro servers. The communication channel of the IKEv2 protocol is protected by the IKEv2 itself. This step establishes the IPsec communication channel and completes the device authentication. Next, the *control point* terminal searches for UPnP-aware devices in the local area network. In the UPnP framework, a user can control UPnP-aware devices using a special terminal called the *control point*. The search process is executed by the UPnP's discovery protocol. The discovery protocol is called SSDP (Simple Service Discovery Protocol). The SSDP utilizes HTTPMU (Hypertext Transfer Protocol Multicast over UDP) and HTTPU (Hypertext Transfer Protocol over UDP). The SSDP does not need protection methd because these messages have to be broadcasted. If the *control point* finds a UPnP-aware device then the *control point* requests the service list on the target UPnP device. The following messages are protected by IPsec. To distinguish between the SSDP messages and other UPnP messages (the control, the subscribed event, the description and the presentation), the demonstration system utilizes the IPsec policy mechanism. Each message is distinguished by the transport protocol. If the transport protocol is TCP then IPsec protects the communication channel. If the transport protocol is UDP then IPsec does not protect the communication channel. Thus, except for the SSDP messages, the UPnP messages can be protected by IPsec. Chapter 5 did not show the details of the authorization procedure except for the concept. The rest of this section presents the details of the authorization procedure.

Figure 58 shows the details of the authorization procedure in the example of the UPnP framework. The demonstration system employs the IR remote control unit to control conventional appliances. This section describes the process flow of the authorization procedure. Before the following processes, the demonstration system assumes that an IKEv2 program completes the inter-device authentication and retrieves the peer device's PKC onto the local disk. The peer device's PKC should be verified by the trust anchor CA's PKC in the authentication step. The peer device's PKC is used to verify the relation between the device's PKC and the ACs in the following step.

1. A UPnP program on the micro server receives a request to control the target appliance. The peer UPnP program also sends the ownership as the AC to the UPnP program.

2. The UPnP program verifies the digital signature of the AC by the trust anchor AA's PKC.

3. The UPnP program also checks the relation between the AC and the peer

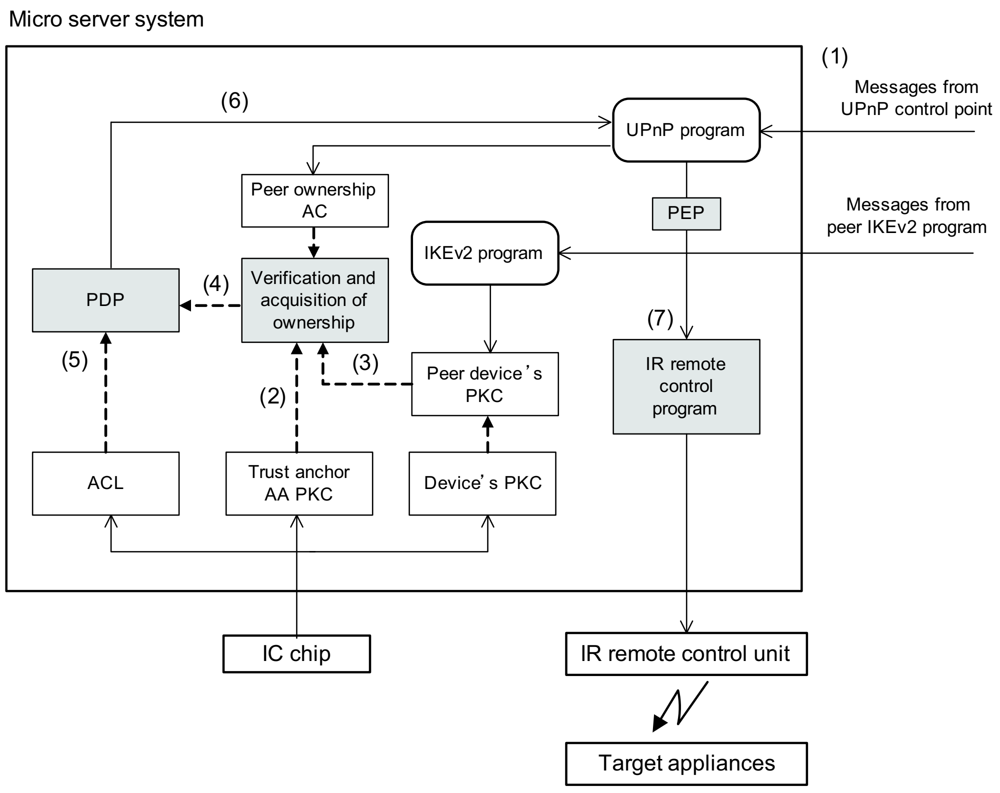


Figure 58. Details of authorization procedure in the example on UPnP framework

device's PKC based on the subject DN field in both certificates.

4. After verification of the AC, the UPnP program retrieves the ownership information (the string in the attribute field) from the AC and passes it to the PDP program.

5. The PDP program decides the authorization result based on the request, the ownership information and the ACL.

6. The PDP program returns the authorization result to the UPnP program.

7. The UPnP program executes the IR remote control program to control the actual appliance's function. If the PDP program returns "Allow" then the PEP function does not do anything. If the PDP program returns "Deny"

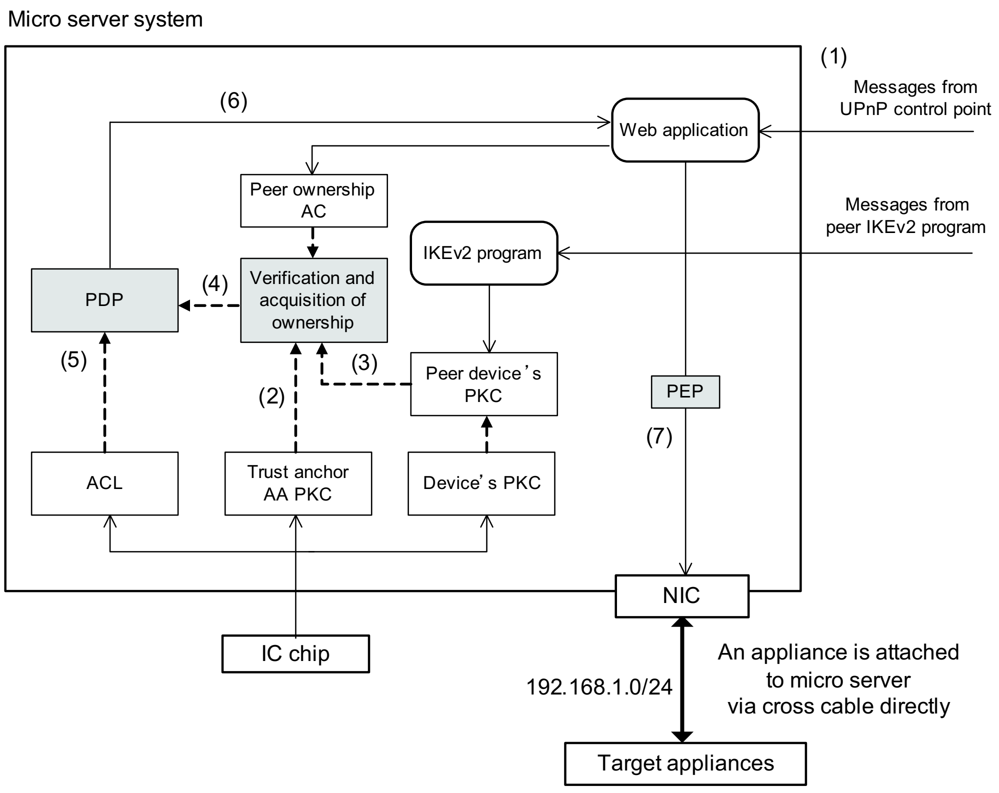


Figure 59. Details of the authorization procedure in the example of IP-based appliances

then the PEP program stops the control procedure and just returns the error code.

In Figure 58, the demonstration system needs to implement the following programs: the PDP program; the verification and acquisition program for the ownership and the IR remote control program. Section 6.4 shows the details of the implementation.

6.3.3 Demonstration System for IP-based Appliances

This section shows the design of the demonstration system for the IP-based appliances. The dissertation calls an appliance with IP-based communication function

Table 16. Hardware specification on the demonstration systems

Micro server	Plathome's Open Micro Server (AMD Alchemy au1550 400MHz)
IR remote control unit	BUFFALO's PC-OP-RS1
IC chip	Gemalto Cyberflex Access e-gate 32k
IC chip reader	Gemalto e-gate token connector

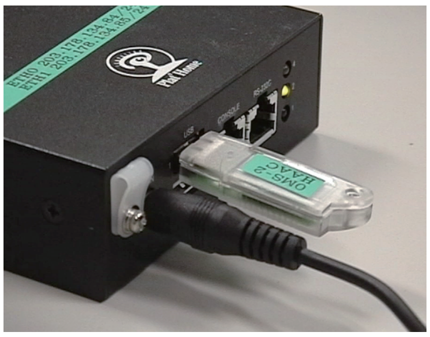


Figure 60. Micro server with proposed IC chip

"IP-based appliance." Figure 59 shows the details of the authorization procedure on the example of the IP-based appliances. The IP-based appliance is attached to the micro server by using a cross cable directly. The micro server this dissertation employs has three physical ports of the Ethernet; therefore, one connector can be used to connect to the Internet, and another connector can be used to connect to the target appliance directly. This example includes a web application. The web application displays the user interface to control the IP-based appliance and it also delegates the request to the IP-based appliances. Thus, the web application works as an application-level proxy program. Most parts of the authorization procedure are the same as in section 6.3.2 except for the web application and the UPnP program.

Table 17. Software specification on the demonstration systems

OS	Debian GNU/Linux 2.6.18
IPv6/IPsec stack	USAGI
IPsec policy management	setkey (ipsec-tools 0.6.6)
UPnP SDK	Intel's portable SDK for UPnP Devices 1.6.0
IKEv2 daemon	racoon2-20070720a
IR remote control	usbserial.ko, ftdi_sio.ko
PC/SC library	PCSC-Lite 1.3.2
CCID driver	ifd-egate-0.05

6.4 Implementation

This section shows the prototype implementation of the demonstration systems shown in section 6.3. This section first presents the hardware specification on the demonstration systems in Table 16. Table 17 shows the software used with the demonstration systems. Figure 60 shows the micro server system with the IC chip and the USB token connector. The dissertation implemented the verification and acquisition program of the ownership and the PDP program as a Perl script. To retrieve the attribute field from the AC, the program has to parse the ASN.1 binary data. The PDP program parses the ACL and decides the authorization result based on the obtained ownership information and the request. The prototype implementation added the above parts to the UPnP program and the web application. The PEP process is also implemented as a function in each program. The UPnP service program is implemented using Intel's SDK shown in Table 17. The IR remote control program utilizes the IR control remote unit (PC-OP-RS1) using the */dev/ttyUSB* device directly. The web application works as an application-level proxy and is implemented by using the PHP script. The web application has the user interface to control the appliance.

6.5 Results in the Demonstration Experiments

The demonstration system deploys the appliances with the micro server and the IC chip in the experimental network. Figure 61 shows the network configuration. This section presents the two kinds of examples. The first example controls the

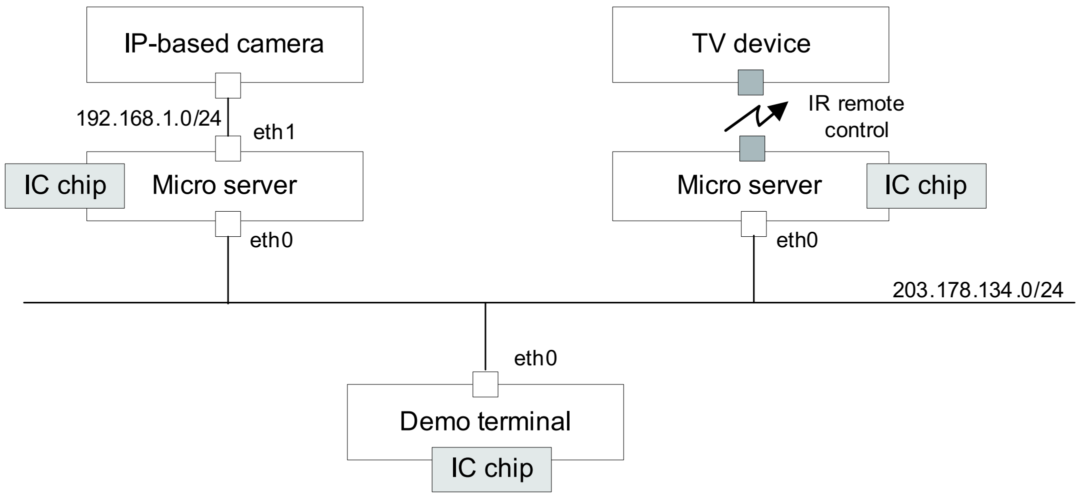


Figure 61. Network configuration for demonstration experiments

TV device, and the second example controls the IP-based security camera. The demonstration experiment also utilizes a special demonstration terminal. The demonstration terminal works as a control device. We assume that the terminal is the a small mobile device such as a PDA or a cell phone. To demonstrate the operation flow on the connecting device, the demonstration utilizes the PC as the demonstration terminal. The demonstration terminal runs the same software with the micro server system.

6.5.1 Preparations for the Demonstration

For the demonstration, we have to set up the IC chip. We can initialize the IC chip using the initialization tool. In the demonstration, the initialization process is done not by the manufacturer but by the demonstrator. Next, we have to set up the ownership information using the personalization tool. When the set up process is completed, we attach the IC chip to the micro server. At this point, the IC chip must have a device's identity (the device's PKC), the ownership (the ACs), the trust anchor CA's PKC, the trust anchor AA's PKC and the ACL. We have to set up three IC chips for the IP-based security camera, the TV device and the demonstration terminal at this time.

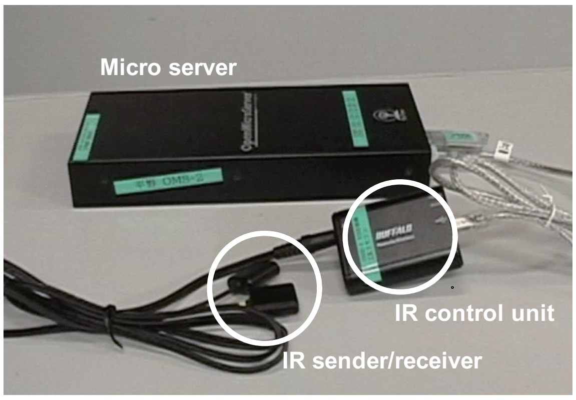


Figure 62. Micro server and IR remote control unit

6.5.2 Demonstration 1: The TV device

Figure 58 shows the demonstration system to control the conventional non-networked home appliance. The micro server for the TV device and the demonstration terminal run the IKEv2 program for inter-device authentication. The micro server for the TV device runs the UPnP service program. The demonstration terminal runs the UPnP *control point* program. The *control point* program is simply modified to transfer the ownership information as the AC. Both UPnP programs are extended as described in section 6.4. Figure 62 shows the micro server system with the IR remote control unit.

Figure 63 shows the demonstration experiments of the TV device. Figure 64 shows the screenshot of the demonstration terminal. The demonstration terminal has the ownership “HiranoPDA” as the AC. The TV device has the ACL shown in Figure 65. The demonstration experiment utilizes the generalized UPnP *control point* software written in Java.

In the demonstration, at first, both IKEv2 programs establish the IPsec connection using each IC chip’s function. Next, the UPnP *control point* program on the demonstration terminal connects to the TV device. The demonstration terminal sends the appropriate AC to the TV device. The UPnP program on the TV side receives the controlling request and executes the access control based on the ACL and received ownership. If the request is not authorized then the UPnP

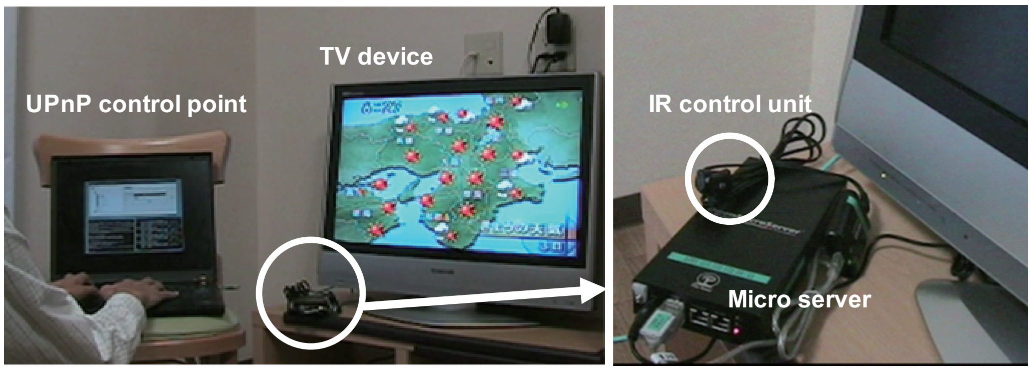


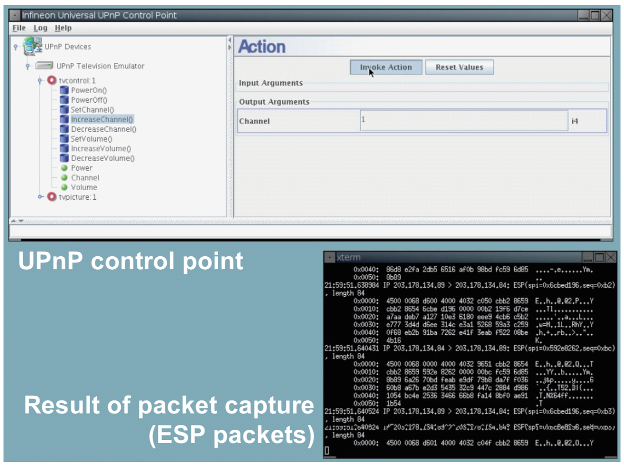
Figure 63. Demonstration experiments of the TV device

service program just returns the error code to the *control point*. If the request is authorized then the UPnP program invokes the IR control program and accesses the actual appliance's functions. The demonstration system can execute the following commands: PowerOn, PowerOff, IncreaseChannel, DecreaseChannel, IncreaseVolume and DecreaseVolume.

6.5.3 Demonstration 2: The IP-based Security Camera

Figure 59 shows another demonstration system to control the IP-based appliance. The demonstration system employs the Panasonic BB-HCM110 security camera. Figure 66 shows the actual demonstration system for the IP-based security camera. The micro server for the security camera and the demonstration terminal run the IKEv2 program for the inter-device authentication. The micro server for the security camera runs the web application. The demonstration terminal runs a regular web browser application.

In the demonstration, at first, the demonstration terminal sends the ownership as the AC to the security camera. Figure 67 shows a screenshot of the demonstration terminal. When the demonstration terminal connects the security camera for the first time, the IKEv2 programs on both micro servers begin to negotiate the CHILD-SA and establish the IPsec connection. Until the IPsec connection is established, the request is blocked. Therefore, at the initial request, the web browser shows the “Connection Interrupted” error. After the establish-



```
        0x0040:  86d8 e2fa 2db5 6516 af0b 98bd fc59 6d85  ....-.e......Ym.
        0x0050:  8b89                                     ..
21:59:51.638984 IP 203.178.134.89 > 203.178.134.84: ESP(spi=0x6cbed196,seq=0xb2)
, length 84
        0x0000:  4500 0068 d600 4000 4032 c050 cbb2 8659  E..h..@.@2.P...Y
        0x0010:  cbb2 8654 6cbe d196 0000 00b2 19f6 d7ce  ...Tl...........
        0x0020:  a7aa deb7 a127 10e3 6180 eee9 4cb6 c5b2  .....'..a...L...
        0x0030:  e777 3d4d d6ee 314c e3a1 5268 59a3 c259  .w=M..1L..RhY..Y
        0x0040:  0f68 eb2b 91ba 7262 e41f 3eab f522 08be  .h.+..rb..>.."..
        0x0050:  4b16                                     K.
21:59:51.640431 IP 203.178.134.84 > 203.178.134.89: ESP(spi=0x592e8262,seq=0xbc)
, length 84
        0x0000:  4500 0068 0000 4000 4032 9651 cbb2 8654  E..h..@.@2.Q...T
        0x0010:  cbb2 8659 592e 8262 0000 00bc fc59 6d85  ...YY..b.....Ym.
        0x0020:  8b89 6a26 70bd feab e9df 79b8 da7f f036  ..j&p.....y....6
        0x0030:  60b8 a67b e2d3 5435 32c9 447c 2884 d986  `..{..T52.D|(...
        0x0040:  1054 bc4e 2536 3466 66b8 fa14 8bf0 ae91  .T.N%64ff.......
        0x0050:  1b54                                     .T
21:59:51.640524 IP 203.178.134.89 > 203.178.134.84: ESP(spi=0x6cbed196,seq=0xb3)
, length 84
21:59:51.640524 IP 203.178.134.89 > 203.178.134.84: ESP(spi=0x6cbed196,seq=0xb3)
, length 84
        0x0000:  4500 0068 d601 4000 4032 c04f cbb2 8659  E..h..@.@2.O...Y
```

Figure 64. Screenshot of the demonstration terminal

ment of the IPsec connection, the web browser can display the user interface of the security camera.

Figure 68 shows the ACL in the security camera. The request from the demonstration terminal with the ownership “HiranoPDA” can complete all requests. However, the request from the demonstration terminal with the ownership “TanakaPDA” can complete only “ViewPicture.” The “TanakaPDA” cannot access other functions on the security camera. Figure 69 shows the screenshot of the result of the access control mechanism. The demonstration system can control the following operations: ViewPicture, TiltUp, TiltDown, PanRight, PanLeft, DefaultBrightness, Darker and Brighter.

6.6 Summary

A traditional security mechanism like TLS is still useful for inter-device communication. However, the characteristics of the inter-device communication require some novel security mechanisms. Therefore, this dissertation has introduced a novel inter-device authentication and authorization framework. This chapter has especially presented the actual security surveillance systems to illustrate the usability of the proposed inter-device authentication and authorization framework.

```
Deny from All;
Allow "PowerOn" from "HiranoPDA";
Allow "PowerOff" from "HiranoPDA";
Allow "IncreaseChannel" from "HiranoPDA";
Allow "DecreaseChannel" from "HiranoPDA";
Allow "IncreaseVolume" from "HiranoPDA";
Allow "DecreaseVolume" from "HiranoPDA";
```

Figure 65. ACL in the TV device

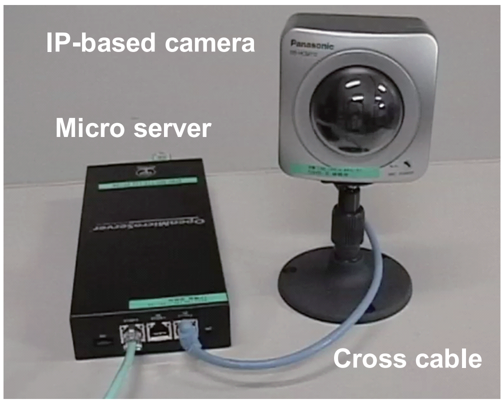
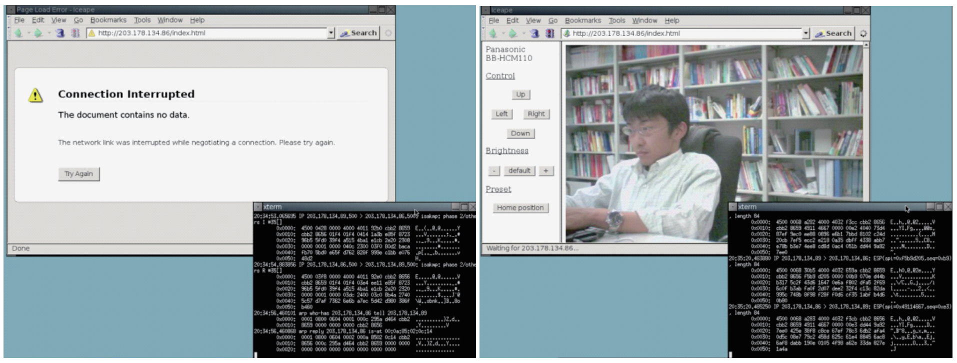


Figure 66. Demonstration system for the IP-based security camera



Connection Interrupted

The document contains no data.

The network link was interrupted while negotiating a connection. Please try again.

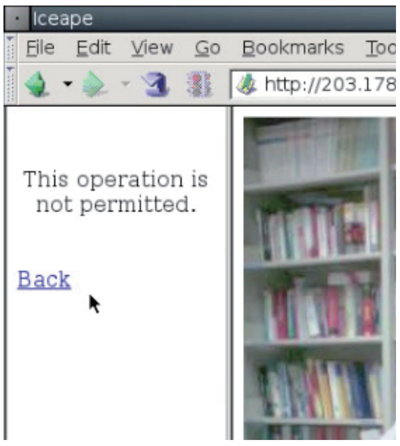
```
20:34:53.065695 IP 203.178.134.89.500 > 203.178.134.86.500: isakmp: phase 2/others I #35[]
        0x0000:  4500 0428 0000 4000 4011 92b0 cbb2 8659  E..(..@.@......Y
        0x0010:  cbb2 8656 01f4 01f4 0414 1a3b e85f 8723  ...V.......;._.#
        0x0020:  96b5 5fd0 39f4 a515 4ba1 e1cb 2e20 2308  .._.9...K.....#.
        0x0030:  0000 0001 0000 040c 2300 03f0 80d2 baca  ........#.......
        0x0040:  fb70 5bd0 e65f d762 828f 999e c1bb e076  .p[.._.b.......v
        0x0050:  48d2                                     H.
20:34:54.883856 IP 203.178.134.86.500 > 203.178.134.89.500: isakmp: phase 2/others R #35[]
        0x0000:  4500 03f8 0000 4000 4011 92e0 cbb2 8656  E.....@.@......V
        0x0010:  cbb2 8659 01f4 01f4 03e4 ee11 e85f 8723  ...Y........._.#
        0x0020:  96b5 5fd0 39f4 a515 4ba1 e1cb 2e20 2320  .._.9...K.....#.
        0x0030:  0000 0001 0000 03dc 2400 03c0 8b4a 2740  ........$....J'@
        0x0040:  5c57 d7af 7362 6e6b a7ec 5d42 d980 386f  \W..sbnk..]B..8o
        0x0050:  b460                                     .
20:34:56.460101 arp who-has 203.178.134.86 tell 203.178.134.89
        0x0000:  0001 0800 0604 0001 000c 295a d464 cbb2  ..........)Z.d..
        0x0010:  8659 0000 0000 0000 cbb2 8656            .Y.........V
20:34:56.460868 arp reply 203.178.134.86 is-at 00:0a:85:02:0c:14
        0x0000:  0001 0800 0604 0002 000a 8502 0c14 cbb2  ..............
        0x0010:  8656 000c 295a d464 cbb2 8659 0000 0000  .V..)Z.d...Y....
        0x0020:  0000 0000 0000 0000 0000 0000 0000       ..............
```

```
, length 84
        0x0000:  4500 0068 a282 4000 4032 f3cc cbb2 8656  E..h..@.@2.....V
        0x0010:  cbb2 8659 4911 4667 0000 00e2 4040 73d4  ...YI.Fg....@@s.
        0x0020:  87ef 9ec0 ee88 0896 e6b1 7bbd 8102 c24d  ..........{....M
        0x0030:  20cb 7ef5 ecc2 e218 0a35 dbff 4338 abb7  ..~......5..C8..
        0x0040:  e78b b3e7 4ee8 cd8d 0ac4 051b dd44 9a92  ....N........D..
        0x0050:  7ee0                                     ~.
20:35:20.483880 IP 203.178.134.89 > 203.178.134.86: ESP(spi=0xf5b9d205,seq=0xb9)
, length 84
        0x0000:  4500 0068 30b5 4000 4032 659a cbb2 8659  E..h0.@.@2e....Y
        0x0010:  cbb2 8656 f5b9 d205 0000 00b9 070e d44b  ...V...........K
        0x0020:  b317 5c2f 43d6 1647 0e6a f802 dfa5 2f69  ..\/C..G.j..../i
        0x0030:  6c0f b3ab fa0f 2d07 dee2 32f4 c13c 82da  l.....-...2..<..
        0x0040:  995c 748b 8f98 f28f f0d6 cf35 1abf b4d6  .\t........5....
        0x0050:  0b80                                     ..
20:35:20.485250 IP 203.178.134.86 > 203.178.134.89: ESP(spi=0x49114667,seq=0xe3)
, length 84
        0x0000:  4500 0068 a283 4000 4032 f3cb cbb2 8656  E..h..@.@2.....V
        0x0010:  cbb2 8659 4911 4667 0000 00e3 dd44 9a92  ...YI.Fg.....D..
        0x0020:  7ee0 425e 38f8 c8ce 67ef 78c3 6db2 afa4  ~.B^8...g.x.m...
        0x0030:  0d5c 08e7 79c2 458d 625c 61e4 8845 6ac8  .\..y.E.b\a..Ej.
        0x0040:  6af8 dabb 190e 0105 4f98 a62e 33da 827e  j.......O...3..~
        0x0050:  1a4a                                     .J
```

Figure 67. Screenshot of the demonstration terminal

```
Deny from All;
Allow ”ViewPicture” from ”HiranoPDA”;
Allow ”ViewPicture” from ”TanakaPDA”;
Allow ”TiltUp” from ”HiranoPDA”;
Allow ”TiltDown” from ”HiranoPDA”;
Allow ”PanRight” from ”HiranoPDA”;
Allow ”PanLeft” from ”HiranoPDA”;
Allow ”DefaultBrightness” from ”HiranoPDA”;
Allow ”Darker” from ”HiranoPDA”;
Allow ”Brighter” from ”HiranoPDA”;
```

Figure 68. ACL in the IP-based security camera



This operation is not permitted.

Figure 69. Result of access control on the IP-based security camera

7. Discussion

7.1 The Improvements

Chapter 4 shows an extension method of a network layer's security mechanism for user applications. The proposed UADB mechanism extends the network layer's security mechanism to handle the *user* entity and the *service* entity for the authentication and authorization. Figure 70 shows the improvements in the proposed UADB mechanism. Conventional IKEv1 program has mutual host-based authentication and host-based VPN functions. Proposed UADB mechanism has extended the following functions: (1) a user authentication mechanism, (2) a *virtual session* mechanism, and (3) a user ID-based authorization mechanism. The UADB mechanism achieves the user authentication function to extend ISAKMP payloads in IKEv1's phase 1 exchange. Figure 71 shows the concept of proposed *virtual session* mechanism. The *virtual session* mechanism enables a mapping between user ID and socket-pairs.

XAUTH (Extended Authentication within IKE) is an expired Internet Draft in IETF. XAUTH mechanism simply adds a user authentication function after conventional IKEv1's phase 1 exchange. XAUTH is typically used for user authentication on the remote access VPN. Although XAUTH cannot distinguish each user on the same host, the proposed UADB mechanism has *virtual session* mechanism. Thus, although XAUTH cannot be used for user applications, proposed UADB mechanism can be used for user applications. IKEv2 specification [112] also describes extended user authentication method using EAP (Extensible Authentication Protocol). EAP also does not distinguish each user on the same host. Therefore, IKEv2 with EAP cannot be used for user applications directly. In summary, IPsec system with the proposed UADB mechanism can support user applications by using its *virtual session* mechanism.

Chapter 5 presents a novel security mechanism for the inter-device communication paradigm using proven network layer's security protocols, IPsec protocol and IKEv1 protocol. To show the usability of the proposal, the dissertation applies proposed inter-device authentication and authorization framework to UPnP framework in chapter 6. Figure 72 shows the improvements in proposed inter-device authentication and authorization framework. The proposal defines *own-*

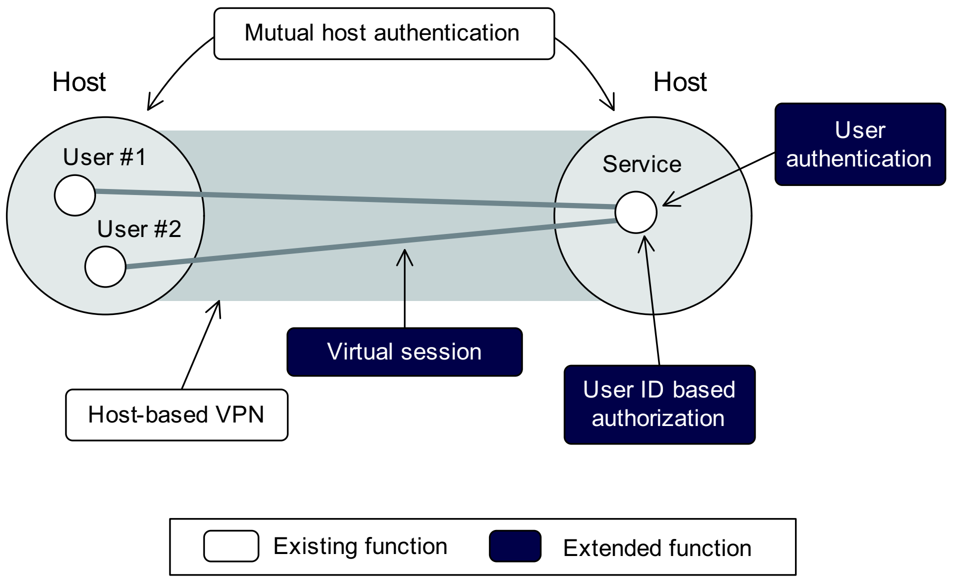


Figure 70. Improvements in proposed UADB mechanism

ership entity associated with *device's ID*. The dissertation expresses the *device's ID* as a public key certificate and the *ownership* as an attribute certificate. The inter-device authentication and authorization framework guarantees the relation between device's ID and ownership based on PKI. Although the inter-device authentication and authorization framework does not support *virtual session* mechanism like UADB mechanism, the proposed framework can achieve authorization based on the *ownership*.

The UADB mechanism adds the *virtual session* mechanism to conventional IPsec system. Therefore, UADB-extended IPsec system can be used as another transport security mechanism like TLS. On the other hand, the inter-device authentication and authorization framework does not add the *virtual session* mechanism. The inter-device authentication and authorization framework leaves the session management to each application's care. However, both mechanisms can achieve the access control based on the user's identity or ownership. As described above, this dissertation has shown two kinds of extension methods of a network layer's security mechanism. Both mechanisms improve granularity of authentication and authorization. Although a conventional IPsec system can

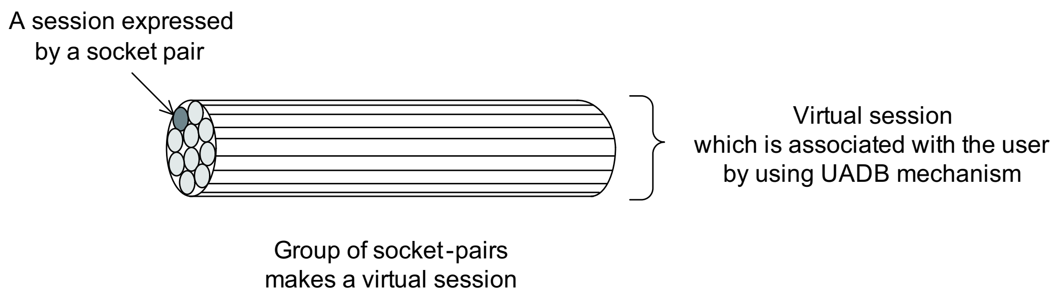

A session expressed by a socket pair
Virtual session which is associated with the user by using UADB mechanism
Group of socket-pairs makes a virtual session

Figure 71. Proposed virtual session mechanism

only handle the *host (i.e. device)* entity, both proposals can additionally handle *user/service* or *ownership* entity. The UADB mechanism binds a *session* and a *user/service*. The inter-device authentication and authorization framework employs typical attribute-based approach using attribute certificate.

Thus, this dissertation has compared two kinds of security mechanisms. The UADB mechanism is well suited to conventional human-controlled devices like PC. However, the inter-device authentication and authorization framework can delegate the *ownership* to the *agent process* on the device. Therefore, the inter-device authentication and authorization framework is well suited to autonomously-controlled devices. The inter-device authentication and authorization framework can meet the requirements for security mechanism on the inter-device communication paradigm shown in section 1.5.

7.2 Life Cycle Management for Device's ID and Ownership

This section discusses life cycle management for a device's ID and ownerships. Figure 73 shows the life cycle of a device. At the production phase, the manufacturer registers the device's ID on the IC chip. The device's ID is represented by a pair of a private key and a PKC. Figure 74 shows the composition of the IC chip. The PKC includes the production number, the model number and the vendor information etc. This information is stored in the Subject DN field of the PKC. The manufacturer can also register some trust anchor CA's PKC on the IC chip in the production phase. The private key and certificates are treated as

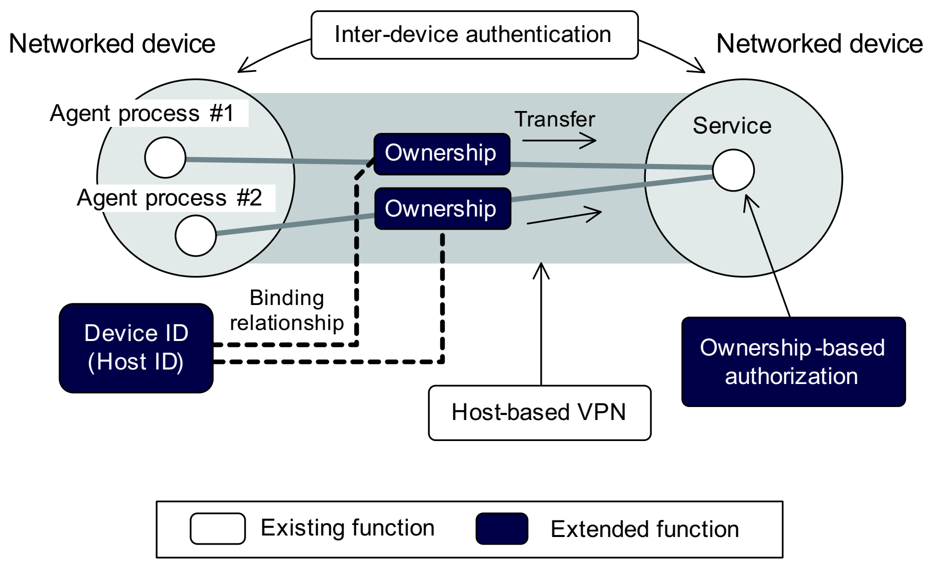


Figure 72. Improvements in proposed inter-device authentication and authorization framework

persistent data, which cannot be changed after the production phase. After the user purchases the product, the user first personalizes the IC chip of the product. At this personalization phase, the user registers the AC as her or his ownership and the ACL on the IC chip. As a result, the product will be able to utilize the ownership in inter-device communication. The ownership and the ACL are user-defined variable data. A typical user's AA issues many ACs because a typical user has many devices. The user can renounce ownership of the intended device to remove her or his AC from the IC chip.

In an actual environment, devices can be leased, resold and transferred. In such a case, previous ownership information that is represented by AC and ACL has to be deleted from the IC chip. If previous owner leaves her or his private information on the IC chip, the next owner can misuse the previous owner's rights. If the owner lends a device such as mobile phone to her or his friend for short time, it might be used with original ownerships. In such case, the friends can use the owner's device with the original rights. As described in section 5.9.1, an *owner PIN* is needed to update ownership and ACL. Therefore, the *owner PIN*

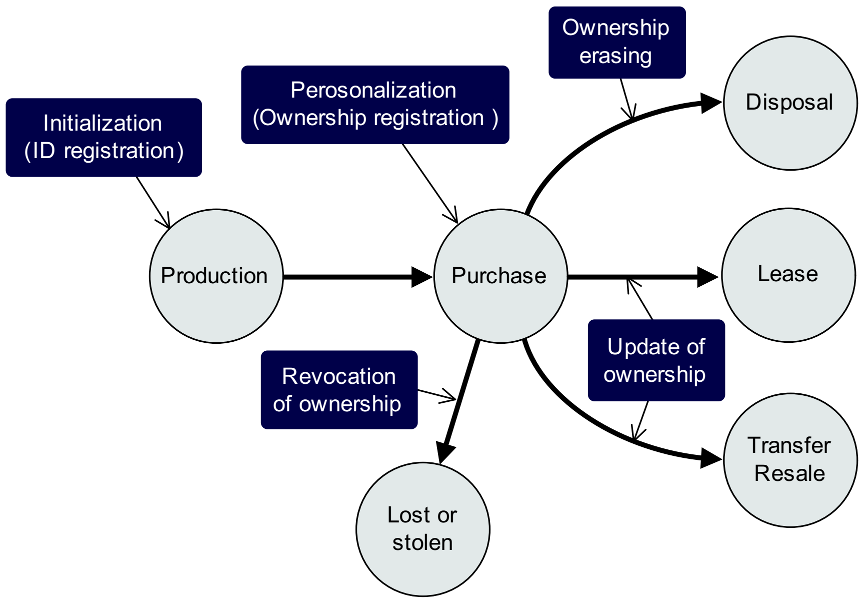


Figure 73. Life cycle management for devices

also has to be initialized for the next owner. At the disposal phase, the last owner has to delete the ownership and ACL to prevent misuse.

If the device is stolen then the user must take one of the following actions: (1) the user's AA issues a new CRL for the stolen device's AC and stores the CRL on all other devices, and (2) the user revokes her or his AA's PKC and makes new AA. The new AA issues all other devices' ACs again and stores them on all devices. A typical AC has a shorter lifetime than a PKC. The dissertation recommends that the user has to make an AC with a short life time of about a few months. As a result, a stolen device can only be used in a limited period thus containing the damage. The CRL for a manufacturer's CA is not needed in normal conditions. However, if a manufacturer's CA is attacked and its private key is stolen then the manufacturer has to issue a new CRL to revoke the CA's PKC. The manufacturer has to update all related devices with the CRL. To prevent all devices' IDs from becoming invalid, the manufacturer should install some backup devices' IDs and trust anchor CA's PKCs issued by different backup CAs. Moreover, to update many IC chips efficiently, a new network-based update

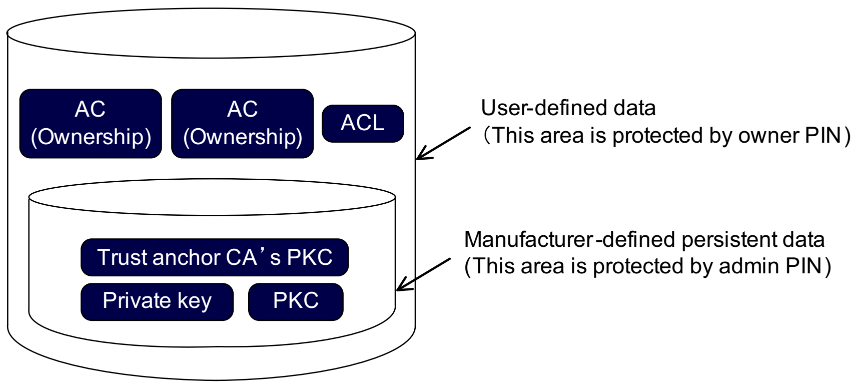

AC (Ownership)
AC (Ownership)
ACL
User-defined data
(This area is protected by owner PIN)
Trust anchor CA's PKC
Private key
PKC
Manufacturer-defined persistent data
(This area is protected by admin PIN)

Figure 74. Device's identity and ownership

mechanism for the IC chip is needed. If a user's AA is attacked and its private key is stolen then the user has to revoke her or his AA's PKC and make a new AA. The new AA issues all devices' ACs again and stores them on all devices.

7.3 Problem of Hardware Performance

This section discusses problems of hardware performance. Okabe et al. [13] show a specialized security mechanism for the BA system. They show performance problems on public key cryptography like RSA for low-performance devices. A typical sensor device in the BA system can not handle public key cryptography with its limited hardware. For example, a current typical sensor device has only 512KB ROM, 512KB RAM and 8-16bit MPU. It does not have a non-symmetric cryptographic coprocessor. They intend to introduce an IP-based control network to a conventional BA system. Therefore, they propose novel security architecture for the BA system based on IPsec and KINK (Kerberized Internet Negotiation of Keys), not IKE. Typical IKE negotiation needs public key cryptography to exchange keys. They show authentication and key exchange methods for IPsec without public key cryptography. BACnet [11] supports symmetric cryptography like DES. However, it does not support non-symmetric key cryptography like RSA. LonWorks [12] supports only simple server authentication. As described above, security mechanisms for the BA system have some problems because of its limited hardware performance. This dissertation shows an inter-device authentication and encrypted communication mechanism based on IPsec and IKE. The dissertation proposes novel software running on the smart card with a non-symmetric cryptographic coprocessor. A tamper-resistant device such as a smart card with a coprocessor is still expensive and it has limited performance for public key encryption. However, it is a reasonable choice to implement public key cryptography on resource-limited embedded devices. However, the proposal is still dependent on the original hardware performance because IKE executes other non-symmetric cryptographic operations like DH (Diffie-Hellman) exchange. On the other hand, Potlapally et al. show a high-performance IPsec execution method on resource-constrained embedded processors [116].

A current typical home appliance has 512KB ROM, 16-32KB RAM and 8-16bit MPU. Most home appliances do not have coprocessors for public key cryptography. Therefore, it has the same performance problem on the BA system. Currently, there are a large number of TRON-based home appliances [117]. Sakamura et al. showed the eTRON architecture for E-commerce in 2001 [118]. They introduce a novel tamper-resistant IC chip for TRON devices, which is used for E-commerce transactions. The eTRON is provided in the form of a smart card or an IC chip with 8-32 bit MPU. The eTRON supports both contact type interfaces defined by ISO7816 and non-contact type interfaces defined by ISO14443. The eTRON also supports PKI functions. However, the eTRON does not support the separation of ownership and a device's ID as this dissertation proposes.

Embedded devices in current vehicles have more powerful and reliable 32 bit RISC processors. These processors can process public key cryptography and other cryptographic functions like symmetric key encryption or hash. Typical vehicles do not have tamper-resistant secure storage, which is used to store the vehicle's ID and other private data. Therefore, future vehicles will need some specialized processors and memory for security purposes.

Other extremely resource-constrained devices such as sensor nodes need other security mechanisms to establish secure communication. Public nodes usually do not need strict security mechanisms. However, some kinds of private nodes need security mechanisms. These resource-constrained devices can employ a proxy-based approach to reduce the workload of cryptographic operations. However, a

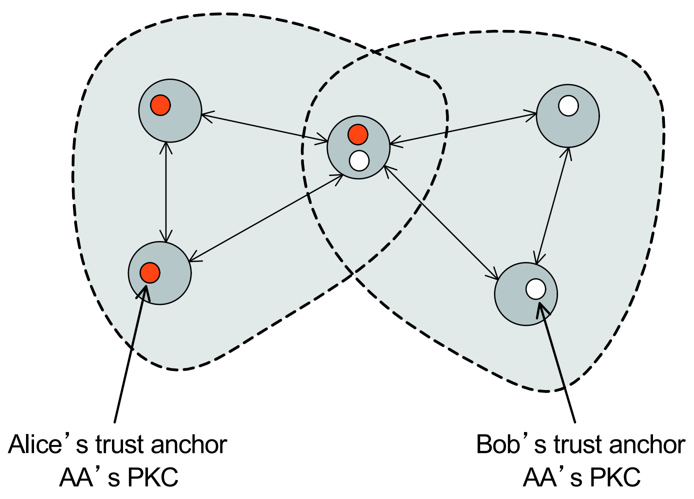


Figure 75. Simple group management method for devices

typical sensor network is not suited for a PKI-based security mechanism because it is difficult to manage a large number of sensor nodes by CA. A sensor network with resource-constrained hardware needs different security mechanisms.

7.4 Collective Operations

This dissertation has shown a novel security mechanism for an inter-device communication paradigm. Figure 75 shows a simple group management method for multiple users' devices. In this example, Alice has three devices with her AA's PKC. Bob also has three devices with his AA's PKC. If one device of them has both AAs' PKCs then the device can be controlled by both Alice's device and Bob's device. This group management method is achieved by the proposed *multiple ownerships model* shown in section 5.4.

Figure 75 shows a more practical group management method based on a tree-structured AA. In this example, there are four users with each trusted domain, which is defined by each user's AA's PKC. If all four users share some public devices, the shared devices have to have all four AAs' PKCs. If the number of users is N then the shared device needs N AAs' PKCs. It is not efficient for most large organizations. To improve its management cost, the proposal can employ

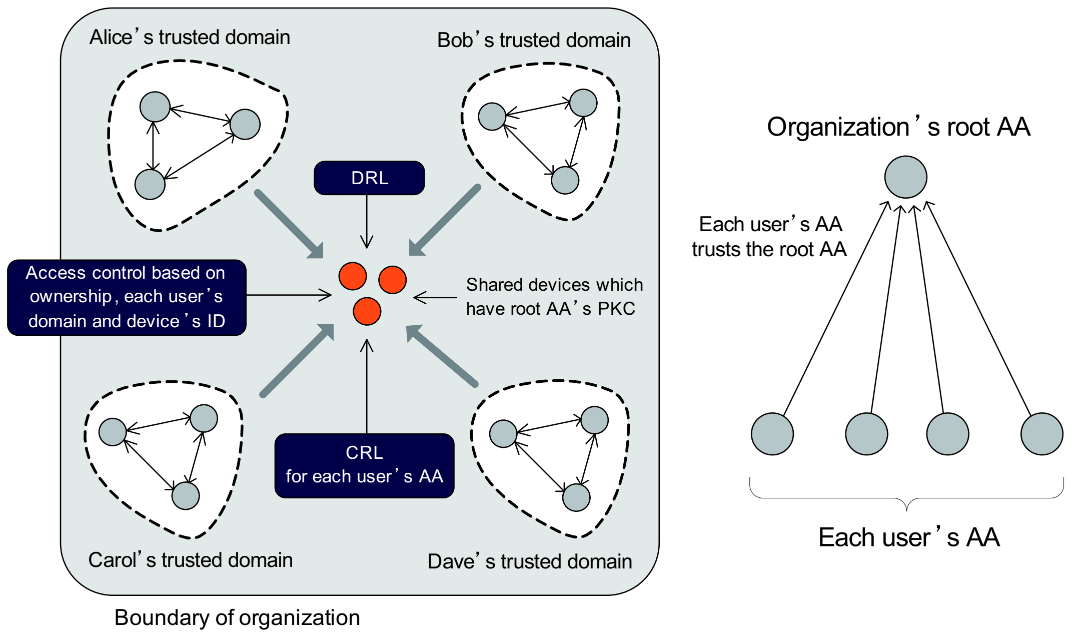


Figure 76. Group management method based on tree-structured AA

a tree-structured AA. The shared devices have to have only the root AA's PKC. If the shared device receives the request from each user's device then the shared device first verifies each received user's AA's PKC by the root AA's PKC. If the verification is successful then the shared device trusts each received user's AA's PKC. Next, the shared device verifies the received AC by the AA's PKC, which is verified in the previous step. If all verifications are successful, then the shared device can trust the received AC's ownership. As a result, the shared device can control requests based on each user's ownership. This group management method also enables authorization based on domain information. For example, the shared device can trust Dave's trusted domain, which includes all of Dave's devices.

To verify tree-structured AAs, the shared device has to obtain the latest CRL from the organization's repository. The CRL can manage the revocation status for each user's AA's PKC. If Alice quits the company then the company's root AA has to issue a new CRL for her AA's PKC. The shared device can also employ a DRL (Device Revocation List), which is issued by the organization's root AA.

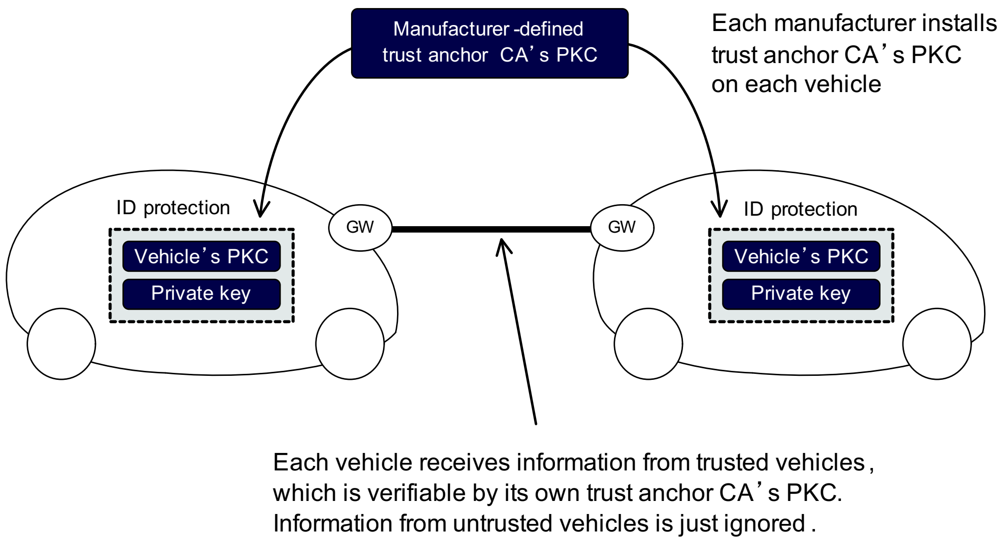

Manufacturer-defined trust anchor CA's PKC
Each manufacturer installs trust anchor CA's PKC on each vehicle
ID protection
Vehicle's PKC
Private key
GW
GW
ID protection
Vehicle's PKC
Private key
Each vehicle receives information from trusted vehicles, which is verifiable by its own trust anchor CA's PKC. Information from untrusted vehicles is just ignored.

Figure 77. Example application in inter-vehicle communication

The DRL can manage a list of devices' IDs to be revoked in the organization. If Bob has his device stolen then he has to declare the device's ID to the organization and the device's ID is listed on the DRL. As mentioned above, the inter-device authentication and authorization framework can be utilized systematically for device group management.

7.5 Example Applications of Inter-device Authentication and Authorization Framework

This dissertation shows a demonstration system for a TV device and security camera in chapter 6. This section shows more practical applications for future inter-device communication. Figure 77 shows an example application in inter-vehicle communication. This example is the most basic usage of inter-device authentication. Each vehicle has its own vehicle's ID (i.e. private key and PKC) and manufacturer's trust anchor CA's PKC. This trust anchor CA's PKC can be installed in the production phase. A vehicle receives information only from trusted vehicles, which are verifiable by the trust anchor CA's PKC. A vehicle does not receive information from untrusted vehicles, which are not verifiable by

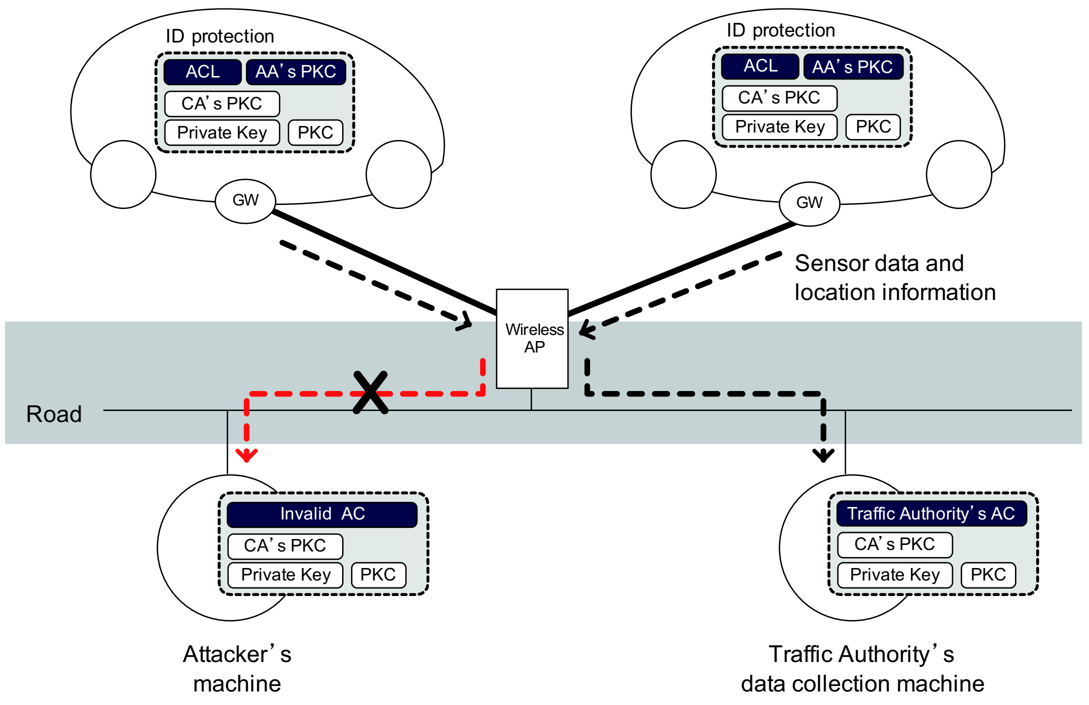


Figure 78. Example application in vehicle's sensor data collection

the trust anchor CA's PKC. To achieve this method securely, the vehicle's ID must be protected from identity-theft by a strong physical security mechanism.

Figure 77 shows an example application for a vehicle's sensor data collection. Each vehicle sends sensor data to a trusted traffic authority machine. First, a vehicle and a traffic authority machine authenticate each other. If both the vehicle and authority machine can verify each ID using each trust anchor CA's PKC, then the vehicle and the machine can establish a trusted encrypted communication channel. Next, the traffic authority machine sends its own AC to the peer vehicle. The vehicle verifies the received AC by the trust anchor AA's PKC. If the verification is successful then the vehicle can authorize the request from the traffic authority machine. If verification fails then the request is just ignored. In this example, each vehicle's owner can protect her or his vehicle's private sensor information from an attacker. If the vehicle's owner does not care about public viewing of its sensor data, the owner just deletes the ACL and trust anchor AA's

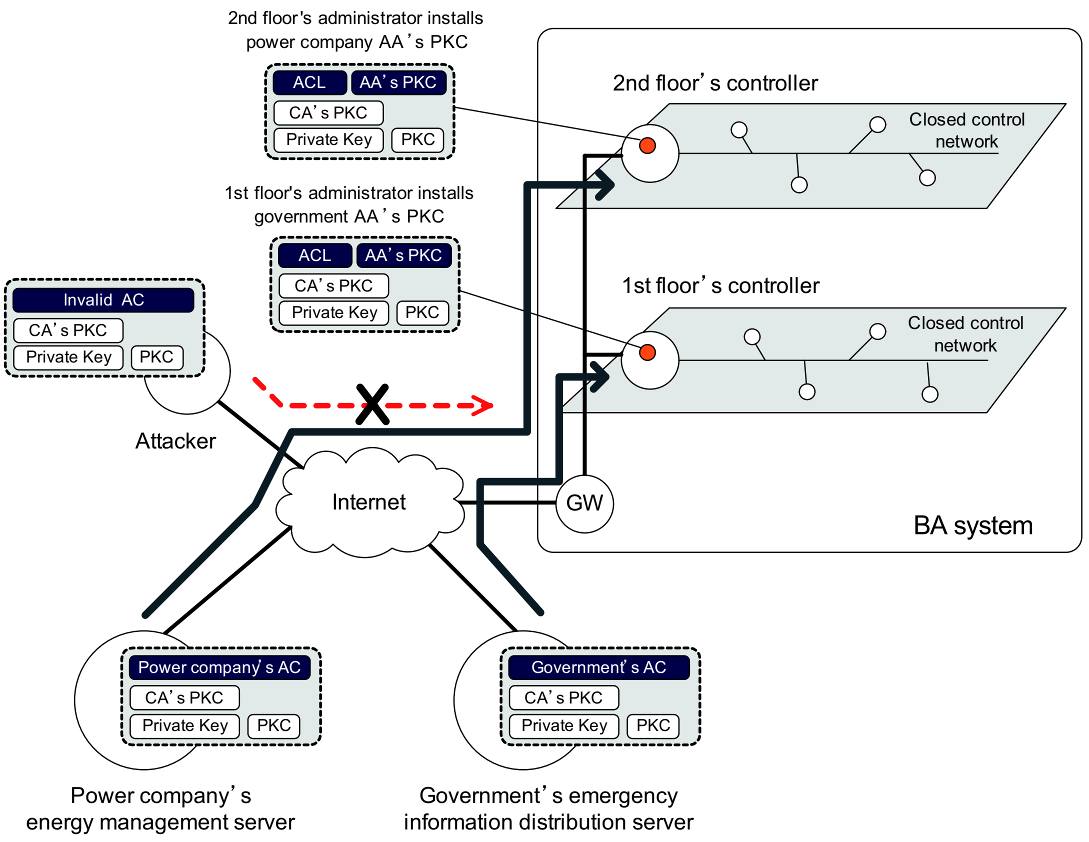


Figure 79. Example application in BA system

PKC issued by the traffic authority.

Figure 77 shows an example application for a BA system. The first floor's administrator installs the AA's PKC issued by the governmental emergency information distribution center on the first floor's controller. If the first floor's controller receives emergency information about an earthquake, for example, then the controller can operates the disaster prevention system automatically. In this case, the controller authorizes the access based on the government-issued AA's PKC. Of course, emergency information is also distributed by broadcast. If the broadcasted information has a digital signature then the controller can verify its validity. In either case, the controller has to keep the trust anchor AA's PKC distributed by the government secure. In the second case, the second floor's ad-

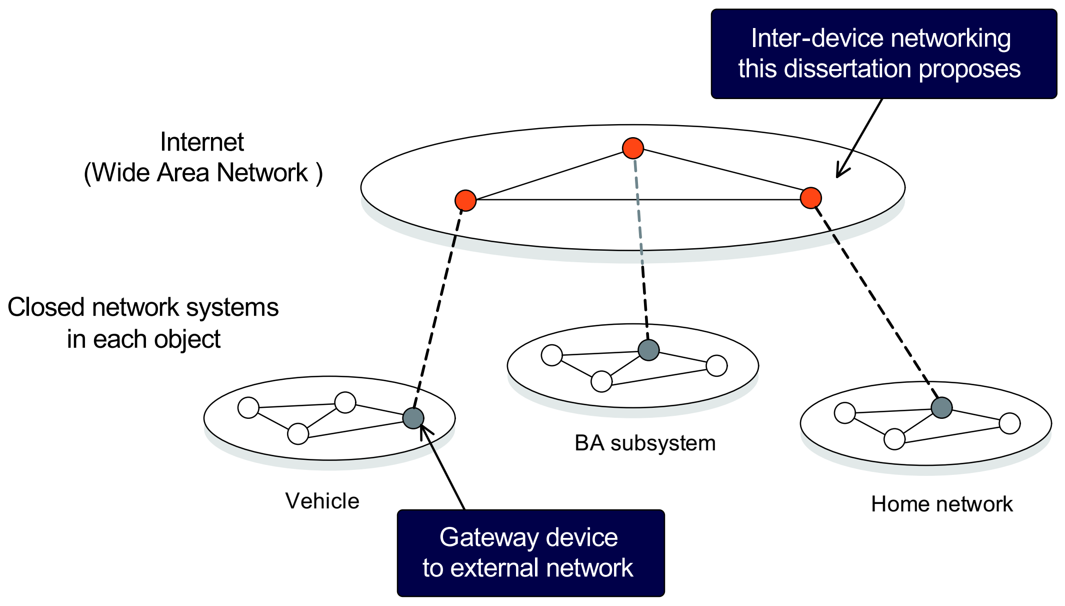


Figure 80. Bridging architecture among heterogeneous networks

ministrator installs the AA's PKC issued by a power company. The office and the power company have a special contract about power consumption. If the power company has to limit the power supply in the summer season, then the power company can send a power-saving request to controllers in contracted offices. The controller can authorize the power-saving request based on the received AC, the power company's AA's PKC and ACL. An attacker cannot access each controller because the attacker's machine does not have a valid AC.

7.6 Example Application among Heterogeneous Systems

Figure 80 shows bridging architecture among heterogeneous systems. Each object such as a vehicle, BA subsystem and home network has its own closed network based on each specific protocol. Each object has at least one gateway device that can connect to the Internet. The gateway devices can make the inter-device networking system this dissertation proposes. Each closed network is constructed on specific protocols like DSRC, BAnet and LonWorks. The gateway device can connect to the Internet and it would employ a conventional TCP/IP system.

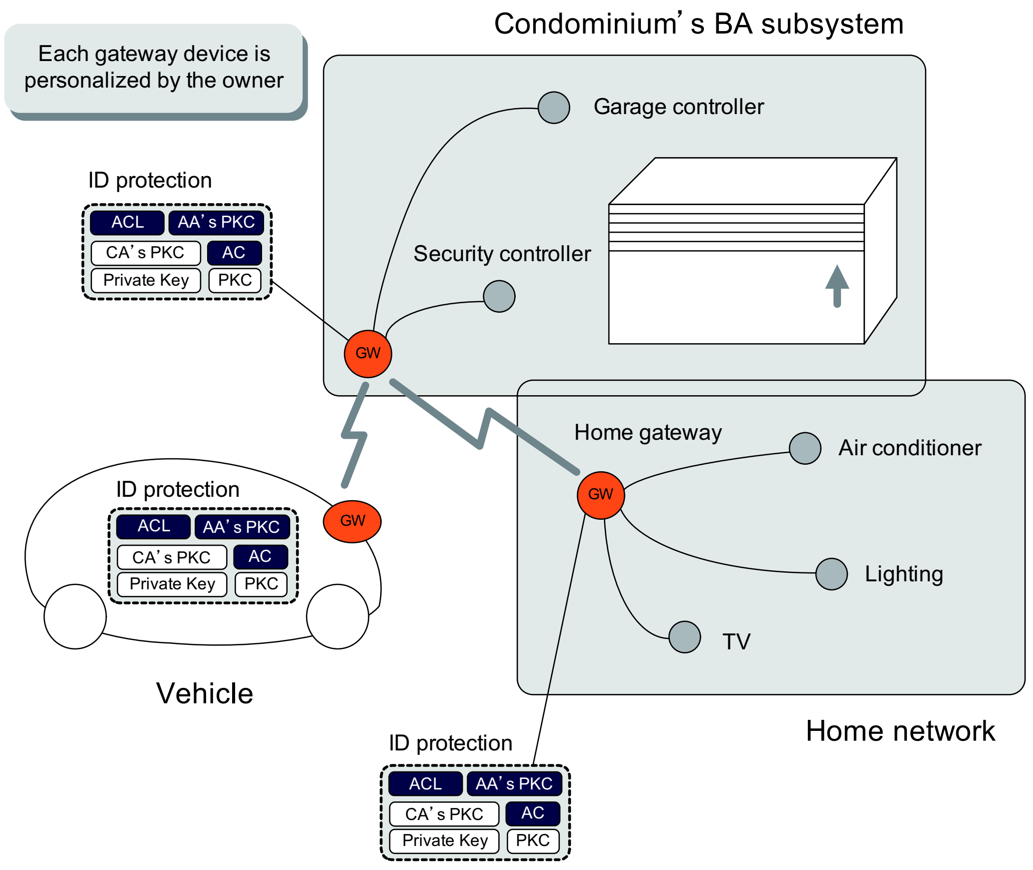


Figure 81. Example application among heterogeneous networks

Figure 81 shows an example application among heterogeneous systems. In this example, all gateway devices of the vehicle, the BA subsystem and the home network have the owner's AC and the owner's AA's PKC. Thus, all devices are personalized by the owner. When the vehicle arrives at home, the condominium's BA subsystem and the vehicle authenticate each other automatically. The BA subsystem authorizes the request from the vehicle based on the received AC. If the request is authorized, then the BA subsystem operates the garage controller to raise the door. The BA subsystem can also change the mode of the security controller. The BA subsystem sends the arrival message to the home gateway. The home gateway and BA gateway authenticate each other automatically. The home gateway can authorize the request from the BA subsystem based on the

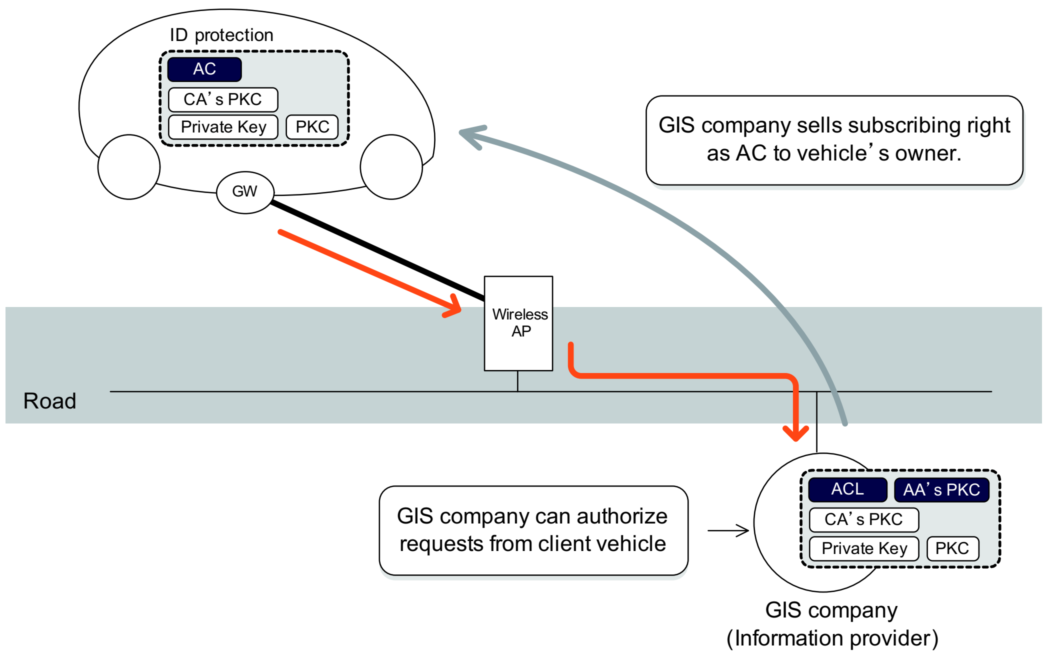

ID protection
AC
CA's PKC
Private Key
PKC
GW
Wireless AP
Road
GIS company sells subscribing right as AC to vehicle's owner.
GIS company can authorize requests from client vehicle
ACL
AA's PKC
CA's PKC
Private Key
PKC
GIS company (Information provider)

Figure 82. Example application for subscribing to GIS information

received AC. When the home gateway trusts the BA subsystem's message, then the home gateway operates air conditioners, lights and the TV device automatically. If the owner has a small wearable wireless controller, then she or he would operate these systems naturally.

7.7 Future Direction

This chapter shows some future directions for the proposed inter-device authentication and authorization mechanism. Future networks will consist of a large number of non-PC networked devices. Although some kinds of distributed devices have limited memory and processors, a large number of distributed devices have the potential to be used for grid computing. These distributed devices also have to have a security mechanism with characteristics suited for grid computing. On the other hand, the proposal can apply to some sort of DRM system. Figure 82 shows the GIS (Geographical Information System) company and the vehicle that is subscribing to the GIS company's data. At first, the vehicle's owner purchases a subscribing right as AC and installs the purchased AC on the vehicle. As a result, the vehicle will be able to access to the GIS company's server. This example employs the AC to transfer the right for the GIS company to the device.

There are many methods to acquire ownership of the device. The simplest method is password-based authentication. A user inputs a password on a device's screen and she or he can acquire ownership of the device. Biometrics is also an effective method to improve security. A wireless wearable token is a more sophisticated method to acquire ownership of the device. The latest vehicles have already employed this kind of wireless ignition key system. This authentication method is based on the user's presence and is a more natural method than inputting a password. On the other hand, there are many authorization methods other than the attribute certificate and the ACL. We also have to consider a more secure and efficient authentication and authorization mechanism for a future inter-device communication system.

7.8 Open Issues

This section summarizes some open issues related to the proposal.

- The dissertation has shown a demonstration system using a micro server with the proposed IC chip. However, we have to downsize the entire system for practical use.

- The current implementation employs a commercially available general-purpose smart card. In future work, we should consider developing specialized embedded hardware with tamper-resistant characteristics.

- We need a remote configuration mechanism for the IC chip. The current implementation requires removing and inserting the IC chip. We should achieve remote update of the ownerships and ACL. We should consider a more efficient and user-friendly mechanism.

- The current implementation only supports a simple syntax of ACL. We can support other security policy mechanisms such as Type Enforcement and

Chinese Wall policy. We should support pluggable access control modules. A policy language like XACML can write more flexible access control rules.

- The PKI issues were described in section 5.12, section 7.2 and section 7.4. We should support multiple manufacturers' CAs and multiple owners' AAs through an efficient method. We have to handle the revocation and recovery of certificates appropriately.
- We should support a group management mechanism for devices. To achieve group management for devices, we can employ both multiple ACs and tree-based AA architecture described in section 7.4.
- The proposal needs a template mechanism to make user-defined access control rules. The manufacturer has to publish an ACL template that defines user-configurable rules, the number of registerable ACs etc.
- For a large number of devices in the organization, the proposal needs a remote management mechanism for distributed devices.
- The current implementation supports only an ownership-based access control mechanism. We have to support a device's ID-based access control. We also have to support access control based on a combination of a device's ID and ownerships.
- This dissertation employs PKC and AC to express a device's ID and ownerships. In future work, we also have to consider another mechanism to achieve the distinguishing and binding of this relation efficiently.

8. Conclusion

This dissertation has first introduced a new paradigm for inter-device communication and discussed security problems that need to be solved. This dissertation has shown the following requirements for the security mechanism of the inter-device communication paradigm: (1) authentication and authorization in a distributed environment, (2) separation between a device's ID and other attributes, (3) association between a device's ID and the attributes, (4) ID protection on distributed devices, and (5) independence from applications and implementations.

This dissertation has mainly presented the following two contributions. First, this dissertation has demonstrated a new attempt to apply the conventional network layer's security mechanism to user-level applications. IPsec protocol is the representative network layer's security mechanism to protect network traffic. IKE protocol (both IKEv1 and IKEv2) works with IPsec and provides automatic key exchange and mutual host authentication. A typical application of the IPsec system is VPN service. The following problems restrict application of IPsec to user-level applications: (1) IKE cannot authenticate a user and a service directly; (2) IPsec cannot map each IP datagram to each user. Therefore, this dissertation has proposed an extension method of a conventional IPsec system for user-level applications. The proposed UADB mechanism especially achieves user authentication other than host authentication of the regular IKE protocol. The UADB mechanism also provides a mapping mechanism between socket-pairs and a user, which is called the *virtual session*. This dissertation has presented the prototype implementation for WWW service to show the usability of the proposal. As a result, the prototype implementation has indicated that the network layer's security mechanism like IPsec can apply to user-level applications. This result has also indicated the feasibility of the encrypted communication and the user authentication for inter-device communication.

Next, this dissertation has presented a novel security model called the *multiple ownerships model*. The proposed model defines a *device's ID* and an *ownership* explicitly. The *device's ID* is expressed by the PKC and the ownership is expressed by the AC. The relation between the *device's ID* and the *ownership* can be verified by the cryptographic techniques of the PKI. This dissertation has also proposed novel smart card software to store the *device's ID* and the *ownerships* securely. This dissertation has shown the design and implementation of smart card software, an initialization tool for manufacturers and a personalization tool for users. The manufacturer can install the *device's ID* on the smart card using the initialization tool. The user can install the *ownership* and *access control list* on the smart card using the personalization tool. The proposal is implemented by a modified IKE program and the proposed smart card software. This dissertation has shown the prototype implementation using both the IKEv1 and IKEv2 software. The proposed system can provide: inter-device authentication based on a device's ID, inter-device authorization based on ownership, traffic confidentiality and integrity service.

To show the usability of the proposed inter-device authentication and authorization mechanism, the dissertation has presented practical demonstration systems. The dissertation has integrated the inter-device authentication and authorization middleware on a micro server. The micro server with the smart card works as the security proxy for the target appliance. The demonstration system can control the devices by the following methods: (1) conventional appliances can be controlled by an IR-remote control unit; (2) appliances with a network function can be controlled by the application-level proxy software and the bridged private network between the target appliance and the micro server. This dissertation has shown demonstration experiments for a TV device and a security camera in an actual environment.

This dissertation has discussed life cycle management for devices, the hardware-dependence problem, group management methods for devices, prospective future applications and bridging architecture among heterogeneous systems.

In summary, this dissertation has demonstrated a new attempt to establish a security mechanism for future networked devices in an inter-device communication paradigm. This dissertation has focused on the authentication and authorization mechanism for the distributed devices and presented a novel security model called the *multiple ownerships model* to distinguish the *device's ID* and the *ownership* explicitly. This dissertation has presented the design and prototype implementation. Finally, this dissertation has shown the usability of the proposal in demonstration experiments.

Acknowledgements

First, I wish to express my gratitude to Professor Suguru Yamaguchi, my advisor and committee chairman, for his support, advice and encouragement. He gave me a great opportunity to research at NAIST. I have learned many things about becoming both a professional researcher and a good educator. Without his kind cooperation, I would not have been able to complete this dissertation.

I would also like to thank Professor Hideki Sunahara for his kind support on my committee. His many valuable comments helped me to finish my dissertation.

I would like to express my appreciation for Professor Yoichi Shinoda at JAIST for his helpful support on my committee. I was able to improve my dissertation with his valuable comments.

I would like to thank associate professor Youki Kadobayashi for his kind support in the IP Lab. He pointed out many important issues for this dissertation. I was able to improve my dissertation through his invaluable comments.

I would like to thank assistant professor Takeshi Okuda and Shigeru Kashihara. Assistant professor Okuda and Taiji Kimura at JPNIC gave me much helpful advice for my papers. I would also like to thank associate professor Eiji Kawai for his support on the Secure VM project. I also thank assistant professor Teruaki Yokoyama at Cyber University and Kenji Masui, Youki Murakoshi and Kazuya Okada at the IP Lab. They helped me to set up my public hearing. I thank Dr. Hiroaki Hazeyama for his useful advice on writing this dissertation. I also thank Daisuke Miyamoto, Yoshihide Matsumoto, Mio Suzuki and Takuji Imura at the IP Lab. I would like to thank all faculty and staff members, current and past members of the Internet Engineering Laboratory (IP Lab) at NAIST. I could not have finished my doctoral work at NAIST without their help and long-lasting friendships during my long graduate school life.

I would like to thank all faculty members in the Department of Information and Computer Engineering, Toyota National College of Technology. I wish to express my gratitude to Professor Fumio Matsuda, Naoki Okabe, Tetsuo Takeshita, Takumi Nakano, Hiroshi Inagaki, Hiroya Andoh, Tsutomu Kimura, Taichi Hayasaka, Nobuyuki Esaki, Atsuhiro Nada and Nobutaka Sasaki. Professor Takeshita gave me useful comments for my research in a special class. All of them gave me the valuable opportunity to get a Ph.D. while working at the college.

I wish to express my gratitude to my former colleagues at the TOSHIBA Corporation. Without their kind cooperation, I could not have completed this dissertation. I would also like to thank Professor Mitaro Namiki at Tokyo University of Agriculture and Technology and Dr. William Saito for his kind support on IPA's project.

Finally, I wish to thank my family. I would like to thank Keiko Nagai and Motoyuki Nagai for their kind support. I thank my sister Keiko and her husband Hiroki. I thank my parents, Toshio and Misako. I thank my wife Yuko. She always supports me and works hard to raise our children. My family's longstanding support has enabled me to complete my degree.

References

[1] Mark Weiser. The computer for the 21st century. *Human-computer interaction: toward the year 2000*, pages 933–940, 1995.

[2] Hiroshi Ishii and Brygg Ullmer. Tangible bits: towards seamless interfaces between people, bits and atoms. In *CHI '97: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 234–241, New York, NY, USA, 1997. ACM.

[3] Fernando J. Corbató, Marjorie Merwin-Daggett, and Robert C. Daley. An experimental time-sharing system. *Classic operating systems: from batch processing to distributed systems*, pages 117–137, 2000.

[4] International Organization for Standardization. ISO11898-1: Road vehicles – Controller area network (CAN) – Part 1: Data link layer and physical signalling, 2003.

[5] LIN Consortium. LIN Specification Package Rev. 2.1, 2006.

[6] MOST Cooperation. MOST specification Rev 2.5, 2006.

[7] IDB Forum. http://www.idbforum.org/.

[8] Hideki SUNAHARA and Keisuke UEHARA. Building the Social System for Automobiles with Internet : Beyond InternetITS to Internet AutoMobility. *IEICE technical report. Information networks*, 102(633):43–46, 20030130.

[9] ARIB Japan. ARIB STD-75: DEDICATED SHORT-RANGE COMMUNICATION SYSTEM ARIB STANDARD, 2001.

[10] ARIB Japan. ARIB STD-T88: DSRC APPLICATION SUB-LAYER ARIB STANDARD, 2004.

[11] International Organization for Standardization. ISO16484: Building automation and control systems – Part 5: Data communication protocol, 2007.

[12] American National Standards Institute. ANSI/EIA 709.1-A-1999 Control Network Protocol Specification, 1999.

[13] Nobuo OKABE, Shouichi SAKANE, Kazunori MIYAZAWA, Kenichi KAMADA, Atsushi INOUE, and Masahiro ISHIYAMA. Security Architecture for Control Networks Using IPsec and KINK. *The transactions of the Institute of Electronics, Information and Communication Engineers. B*, 88(10):1910–1921, 20051001.

[14] ECHONET Consortium. The ECHONET Specification, Version 3.2.1, 2005.

[15] UPnP Forum. UPnP Device Architecture 1.0, July 2006.

[16] Digial Living Network Alliance. DLNA Overview and Vision Whitepaper 2006, 2006.

[17] J. Kohl and C. Neuman. RFC 1510: The Kerberos Network Authentication Service (V5), September 1993.

[18] S. Miller, C. Neuman, J. Schiller, and J. Saltzer. Section E.2.1: Kerberos Authentication and Authorization System. *M.I.T. Project Athena, Cambridge, Massachusetts, December 21*, 1987.

[19] R. Housley, W. Polk, W. Ford, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC3280, 2002.

[20] ITU-T Recommendation X.509 (1997E). Information Technology - Open Systems Interconnection. The Directory: Authentication Framework, June 1997.

[21] Johannes Meinecke and Martin Gaedke. Modeling Federations of Web Applications with WAM. In *LA-WEB '05: Proceedings of the Third Latin American Web Congress*, page 23, Washington, DC, USA, 2005. IEEE Computer Society.

[22] Johannes Meinecke, Martin Gaedke, Frederic Majer, and Alexander Brändle. Capturing the essentials of federated systems. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 895–896, New York, NY, USA, 2006. ACM.

[23] W3C Working Group Note. Web Service Architecture, 2004.

[24] Martin Gaedke, Johannes Meinecke, and Andreas Heil. FDX: federating devices and web applications. In *ICWE*, pages 95–102, 2006.

[25] Hiroshi Igaki, Masahide Nakamura, Haruki Tamada, and Ken ichi Matsumoto. Implementing Integrated Services of Networked Home Appliances Using Service-oriented Architecture. *IPSJ Journal*, 46:314–326, 2005.

[26] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP, 1999.

[27] Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Frystyk Nielsen, Satish Thatte, and Dave Winer. Simple Object Access Protocol (SOAP) 1.1. W3C, 2000.

[28] R. Droms. Dynamic Host Configuration Protocol. RFC2131, 1997.

[29] IANA. Special-Use IPv4 Addresses. RFC3330, 2002.

[30] Sun microsystems. Jini Architectural Overview - Technical White Paper, 1999.

[31] HAVi Organization. HAVi, the A/V digital network revolution, 2006.

[32] Object Management Group. CORBA/IIOP Specification 3.0.3, 2004.

[33] Microsoft. .NET Framework Developer's Guide: .NET Remoting Overview, 2007.

[34] OASIS Open. Web Service Security: SOAP Message Security 1.1, 2006.

[35] Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[36] OASIS Open. Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0, 2005.

[37] OASIS Open. eXtensible Access Control Markup Language (XACML) Version 2.0, 2005.

[38] Carl Ellison. UPnP Security Ceremonies Design Document For UPnP Device Architecture 1.0, 2003.

[39] ISO/IEC. ISO/IEC 7498-2: information processing systems - open systems interconnection reference model - part 2: Security architecture, 1989.

[40] ITU. ITU X.800: Security Architecture for Open Systems Interconnection for CCITT Applications, 1991.

[41] U.S. Department of Defence. Trusted Computer System Evaluation Criteria. Standard DoD 5200.28-STD, 1985.

[42] Comission of the European Communities. Information Technology Security Evaluation Criteria, version 1.2, Directorate General XIII, 1991.

[43] ISO/IEC 15408. Information technology – Security techniques – Evaluation criteria for IT security, 1999.

[44] Audun Josang, John Fabre, Brian Hay, James Dalziel, and Simon Pope. Trust requirements in identity management. In *ACSW Frontiers '05: Proceedings of the 2005 Australasian workshop on Grid computing and e-research*, pages 99–108, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc.

[45] Swapnil Pathare and Sukumar Nandi. Sahnet: a secure system for ad-hoc networking using ecc. In *Q2SWinet '07: Proceedings of the 3rd ACM workshop on QoS and security for wireless and mobile networks*, pages 130–135, New York, NY, USA, 2007. ACM.

[46] M. Burnside, D. Clarke, T. Mills, A. Maywah, S. Devadas, and R. Rivest. Proxy-based security protocols in networked mobile devices. In *SAC '02: Proceedings of the 2002 ACM symposium on Applied computing*, pages 265–272, New York, NY, USA, 2002. ACM.

[47] C. Ellison. SPKI Requirements. RFC2692, 1999.

[48] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. SPKI Certificate Theory. RFC2693, 1999.

[49] Roland Erber, Christian Schlager, and Gunther Pernul. Patterns for Authentication and Authorisation Infrastructures. *dexa*, 0:755–759, 2007.

[50] Reiner Sailer and James R. Giles. Pervasive Authentication Domains for Automatic Pervasive Device Authorization. In *PERCOMW '04: Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, page 144, Washington, DC, USA, 2004. IEEE Computer Society.

[51] Deqing Zou, Laurence T. Yang, Weizhong Qiang, Xueguang Chen, and Zongfen Han. An Authentication and Access Control Framework for Group Communication Systems in Grid Environment. In *AINA '07: Proceedings of the 21st International Conference on Advanced Networking and Applications*, pages 547–554, Washington, DC, USA, 2007. IEEE Computer Society.

[52] D. Ferraiolo and R. Kuhn. Role-Based Access Controls. In *15th NIST-NCSC National Computer Security Conference*, pages 554–563, 1992.

[53] Srdjan Čapkun and Mario Čagalj. Integrity regions: authentication through presence in wireless networks. In *WiSe '06: Proceedings of the 5th ACM workshop on Wireless security*, pages 1–10, New York, NY, USA, 2006. ACM.

[54] Munirul M. Haque, Sheikh I. Ahamed, Haifeng Li, and KM Asif. An Authentication based Lightweight Device Discovery (ALDD) Model for Pervasive Computing Environment. In *COMPSAC '07: Proceedings of the 31st Annual International Computer Software and Applications Conference - Vol. 1- (COMPSAC 2007)*, pages 57–64, Washington, DC, USA, 2007. IEEE Computer Society.

[55] Ron Shacham, Henning Schulzrinne, Srisakul Thakolsri, and Wolfgang Kellerer. Ubiquitous device personalization and use: The next generation of IP multimedia communications. *ACM Trans. Multimedia Comput. Commun. Appl.*, 3(2):12, 2007.

[56] Edgar Weippl and Wolfgang Essmayr. Personal trusted devices for web services: revisiting multilevel security. *Mob. Netw. Appl.*, 8(2):151–157, 2003.

[57] D. E. Bell and L. J. LaPadula. Secure computer systems: A mathematical model. Hanscom AFB, Bedford, MA, Rep. ESD-TR-73-278, vol. 2, ESD/AFSC, 1973.

[58] Fabio Dellutri, Gianluigi Me, and Maurizio A. Strangio. Local Authentication with Bluetooth enabled Mobile Devices. *icas-icns*, 0:72, 2005.

[59] Enrique Soriano Salvador. SHAD: a human centered security architecture for partitionable, dynamic and heterogeneous distributed systems. In *DSM '04: Proceedings of the 1st international doctoral symposium on Middleware*, pages 294–298, New York, NY, USA, 2004. ACM.

[60] S. M. Bellovin. Security problems in the TCP/IP protocol suite. *Computer Communications Review*, 19:2:32–48,http://www.research.att.com/s̃mb/papers/ipext.pdf, 1989.

[61] N. Ferguson and B. Schneier. A Cryptographic Evaluation of IPsec, 2000.

[62] S. Corson and J. Macker. RFC 2501: Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations, January 1999.

[63] Ed C. Perkins. RFC 3344: IP Mobility Support for IPv4, August 2002.

[64] D. Johnson, C. Perkins, and J. Arkko. RFC 3775: Mobility Support in IPv6, June 2004.

[65] V. Devarapalli, R. Wakikawa, A. Petrescu, and P. Thubert. RFC 3963: Network Mobility (NEMO) Basic Support Protocol, January 2005.

[66] Irfan Ahmed, Usman Tariq, Shoaib Mukhtar, Kyung suk Lhee, S.W. Yoo, Piao Yanji, and ManPyo Hong. Binding Update Authentication Scheme for Mobile IPv6. *ias*, 0:109–114, 2007.

[67] André Weimerskirch and Dirk Westhoff. Identity certified authentication for ad-hoc networks. In *SASN '03: Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks*, pages 33–40, New York, NY, USA, 2003. ACM.

[68] Tal Garfinkel, Ben Pfaff, Jim Chow, Mendel Rosenblum, and Dan Boneh. Terra: a virtual machine-based platform for trusted computing. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 193–206, New York, NY, USA, 2003. ACM.

[69] Kurt Dietrich. An integrated architecture for trusted computing for java enabled embedded devices. In *STC '07: Proceedings of the 2007 ACM workshop on Scalable trusted computing*, pages 2–6, New York, NY, USA, 2007. ACM.

[70] Trusted Computing Group. Trusted Platform Modules Strengthen User and Platform Authenticity, 2005.

[71] M. Blaze. Key Management in an Encrypting File System. In *Proceedings of the USENIX Summer 1994 Technical Conference*, pages 27–35, Boston, MA, USA, 6–10 1994.

[72] Ethan L. Miller, William E. Freeman, Darrell D. E. Long, and Benjamin C. Reed. Strong Security for Network-Attached Storage. In *USENIX Conference on File and Storage Technologies (FAST)*, pages 1–14, January 2002.

[73] Vishal Kher and Yongdae Kim. Securing distributed storage: challenges, techniques, and systems. In *StorageSS '05: Proceedings of the 2005 ACM workshop on Storage security and survivability*, pages 9–25, New York, NY, USA, 2005. ACM.

[74] Alan Harbitter and Daniel A. Menascé. The performance of public key-enabled kerberos authentication in mobile computing applications. In *CCS*

'01: Proceedings of the 8th ACM conference on Computer and Communications Security, pages 78–85, New York, NY, USA, 2001. ACM.

[75] Armando Fox and Steven D. Gribble. Security on the move: indirect authentication using Kerberos. In *MobiCom '96: Proceedings of the 2nd annual international conference on Mobile computing and networking*, pages 155–164, New York, NY, USA, 1996. ACM.

[76] Hailiang Mei and Johan Lukkien. A remote personal device management framework based on SyncML DM specifications. In *MDM '05: Proceedings of the 6th international conference on Mobile data management*, pages 185–191, New York, NY, USA, 2005. ACM.

[77] D. Harrington, R. Presuhn, B. Wijnen, and . An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks. RFC3411, 2002.

[78] SyncML Initiative. SyncML V1.0.1 Specification, 2001.

[79] Ian T. Foster, Carl Kesselman, Gene Tsudik, and Steven Tuecke. A Security Architecture for Computational Grids. In *ACM Conference on Computer and Communications Security*, pages 83–92, 1998.

[80] 1 Zhao, 1 Aggarwal, and 1 Kent. PKI-Based Authentication Mechanisms in Grid Systems. *nas*, 00:83–90, 2007.

[81] Bogdan C. Popescu, Bruno Crispo, Andrew S. Tanenbaum, and Frank L.A.J. Kamperman. A DRM security architecture for home networks. In *DRM '04: Proceedings of the 4th ACM workshop on Digital rights management*, pages 1–10, New York, NY, USA, 2004. ACM.

[82] Imad M. Abbadi and Chris J. Mitchell. Digital rights management using a mobile phone. In *ICEC '07: Proceedings of the ninth international conference on Electronic commerce*, pages 185–194, New York, NY, USA, 2007. ACM.

[83] Anthony J. Nicholson, Mark D. Corner, and Brian D. Noble. Mobile Device Security Using Transient Authentication. *IEEE Transactions on Mobile Computing*, 5(11):1489–1502, 2006.

[84] David Jea, Ian Yap, and Mani B. Srivastava. Context-aware access to public shared devices. In *HealthNet '07: Proceedings of the 1st ACM SIGMOBILE international workshop on Systems and networking support for healthcare and assisted living environments*, pages 13–18, New York, NY, USA, 2007. ACM.

[85] Massimo Ficco, Maurizio DfArienzo, and Gianni DfAngelo. A Bluetooth Infrastructure for Automatic Services Access in Ubiquitous and Nomadic Computing Environments. In *MobiWac 07*. ACM, 2007.

[86] Jian Tang, Vagan Terziyan, and Jari Veijalainen. Distributed PIN verification scheme for improving security of mobile devices. *Mob. Netw. Appl.*, 8(2):159–175, 2003.

[87] Mlakar, Zaletelj, and Tasic. Viewer authentication for personalized iTV services. *wiamis*, 00:63, 2007.

[88] S. Kent and K. Seo. Security Architecture for the Internet Protocol. RFC4301, 2005.

[89] T. Dierks and C. Allen. RFC 4346: The Transport Layer Security (TLS) Protocol Version 1.1, April 2006.

[90] Henry M. Levy. Capability-Based Computer Systems, Digital Equipment Corporation, 1984.

[91] D. Harkins and D. Carrel. RFC 2409: The Internet Key Exchange (IKE), November 1998.

[92] B. Gleeson, A. Lin, J. Heinanen, G. Armitage, and A. Malis. RFC 2764: A Framework for IP Based Virtual Private Networks, February 2000.

[93] S. Kent. RFC 4302: IP Authentication Header, December 2005.

[94] S. Kent. RFC 4303: IP Encapsulating Security Payload (ESP), December 2005.

[95] S. Deering and R. Hinden. RFC 2460: Internet Protocol, Version 6 (IPv6) Specification, December 1998.

[96] T. Jinmei, K. Yamamoto, J. Hagino, M. Sumikawa, Y. Inoue, K. Sugyo, and S. Sakane. An Overview of the KAME Network Software: Design and implementation of the advanced internetworking platform. *INET'99*, June 1999.

[97] WIDE Project. http://www.wide.ad.jp/.

[98] D. McDonald, C. Metz, and B. Phan. RFC 2367: PF_KEY Key Management API, version 2, July 1998.

[99] D. Piper. RFC 2407: The Internet IP security domain of interpretation for ISAKMP, November 1998.

[100] D. Maughan, M. Schertler, M. Schneider, and J. Turner. RFC 2408: Internet Security Association and Key Management Protocol (ISAKMP), November 1998.

[101] Bluetooth SIG. Specification of the Bluetooth System Version 2.0 + EDR, 2004.

[102] RSA Laboratories. PKCS11 Cryptographic Token Interface Standard, Version 2.20, 2004.

[103] Sun Microsystems. Java Card 2.1.1 Specifications, 2000.

[104] R. Mayrhofer. Towards an Open Source Toolkit for Ubiquitous Device Authentication. In *Proc. PerSec 2007: 4th IEEE International Workshop on Pervasive Computing and Communication Security*, March 2007.

[105] Khanh V.Nguyen. Simplifying Peer-to-Peer Device Authentication Using Identity-Based Cryptography. In *ICNS '06: Proceedings of the International conference on Networking and Services*, page 43. IEEE Computer Society, 2006.

[106] Fuji Xerox. Introduction on PKI Technology and Device Certificate, 2005.

[107] Anthony J. Nicholson, Mark D. Corner, and Brian D. Noble. Mobile Device Security Using Transient Authentication. *IEEE Transactions on Mobile Computing*, 5(11):1489–1502, 2006.

[108] S. Farrell and R. Housley. An Internet Attribute Certificate Profile for Authorization. RFC3281, 2002.

[109] International Organization for Standardization. *ISO/IEC 7816-4 Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange*, 2005.

[110] Mitsuru Kanda, Kazunori Miyazawa, and Hiroshi Esaki. USAGI IPv6 IPsec Development for Linux. In *Proceedings of the 2004 Symposium on Applications and the Internet-Workshops (SAINT 2004 Workshops)*, page 159. IEEE Computer Society, 2004.

[111] IPsec-Tools. http://ipsec-tools.sourceforge.net/.

[112] C. Kaufman, Ed. Internet Key Exchange (IKEv2) Protocol. RFC4306, 2005.

[113] PCSC-Lite. http://pcsclite.alioth.debian.org/.

[114] VeriSign Device Certificate Service.
http://www.verisign.com/products-services/security-services/pki/device-certificate-service/index.html.

[115] Charlie Kaufman Radia Perlman. Key Exchange in IPSec: Analysis of IKE. *IEEE Internet Computing vol.04 no.6*, 2000.

[116] Anand Raghunathan Nachiketh R. Potlapally, Srivaths Ravi and Ruby B. Lee. Impact of configurability and extensibility on ipsec protocol execution on embedded processors. In *Proceedings of the 19th International Conference on VLSI Design (VLSIDf06)*. IEEE, 2006.

[117] H. Takada and K. Sakamura. Compact, low-cost, but real-time distributed computing for computer augmented environments. *ftdcs*, 00:0056, 1995.

[118] Noboru Koshizuka Ken Sakamura. The eTRON Wide-Area Distributed-System Architecture For E-Commerce. *IEEE MICRO, Vol 21, No.6*, dec 2001.

Appendix

A. List of Publications

A.1 Journal

1. ”Design and Implementation of an Inter-Device Authentication Framework Guaranteeing Explicit Ownership”, Manabu Hirano, Takeshi Okuda, Suguru Yamaguchi, IPSJ Journal, Vol.49, No.2, 2008.

2. ”IPsec と IKE を用いたユーザアクセス制御の枠組みの提案と WWW サーバへの適用”, 平野学, 木村泰司, 山口英, 情報処理学会論文誌, 第 44 巻, 9 号, pp.2344-2352, 2003. (in Japanese)

A.2 International Conference

1. ”Towards Securing Inter-device Communication: Applying Inter-device Authentication and Authorization Framework to Home Appliances”, Manabu Hirano, Takeshi Okuda, Suguru Yamaguchi,23rd Annual Computer Security Applications Conference (ACSAC), Work in Progress Session, Miami FL, Dec. 2007.

2. ”Design and Implementation of a Portable ID Management Framework for a Secure Virtual Machine Monitor”, Manabu Hirano, Takeshi Okuda, Eiji Kawai, Suguru Yamaguchi, 2nd International Workshop on Secure Information Systems (SIS’07), pp.691-700, Wisla, Poland, Oct. 2007.

3. ”Application for a Simple Device Authentication Framework: Device Authentication Middleware using Novel Smart Card Software”, Manabu Hirano, Takeshi Okuda, Suguru Yamaguchi, Proceedings of the International Symposium on Applications and the Internet (SAINT) 2007 Workshops ,IEEE/IPSJ, Hiroshima, Japan, Jan. 2007.

4. ”A Proposal for a Simple Device Authentication Framework: Design and Implementation of Novel Smart Card Software and Its Tools”, Manabu Hirano, Taiji Kimura, Takeshi Okuda,Suguru Yamaguchi, The IET International

Conference on Wireless, Mobile and Multimedia Networks (ICWMMN2006), pp1700-1702, Hangzhou, China, Nov.2006.

A.3 Technical Report (in Japanese)

1. ”仮想計算機環境におけるネットワークアクセス制御層の提案”, 諏訪 公洋, 平野 学, 奥田 剛, 河合 栄治, 山口 英, コンピュータセキュリティシンポジウム 2007 (CSS 2007), Oct. 2007.

2. ”パケットキャプチャデータの効率的な解析を目的とした検索システムの設計と実装”, 加藤 伸哉, 平野 学, マルチメディア, 分散, 協調とモバイル (DICOMO2006) シンポジウム, 2006.

3. ”スマートカードを用いたIPセキュリティポリシと認証情報の配布機構”, 平野学, 山口英, 信学技報（情報セキュリティ研究会）, 105 巻, ISEC-290 号, pp.31-36, 2005.

4. ”ウェブサーバにおけるIPsecとIKEを用いたユーザアクセス制御方式の提案と実装”, 平野学, 木村泰司, 山口英, 情報処理学会, 火の国情報シンポジウム予稿集, pp.353-360, 2002.

A.4 Technical Report (in Japanese; Educational Activity in Toyota National College of Technology since 2005 to 2008)

1. ”公開鍵基盤に基づくロボットの認証及びアクセス制御手法の提案”, 三宅一正, 平野学, 第6回情報科学技術フォーラム (FIT2007), pp.9-10, 2007.

2. ”メール送信ワンタイムパスワードを用いた安全なログインシステムの開発”, 梅田知宏, 平野学, 第6回情報科学技術フォーラム (FIT2007), pp.127-128, 2007.

3. ”P2P技術を用いたウェブサーバの機能拡張”, 松本憲彦, 平野学, 電気関係学会東海支部連合大会, 2007.

4. ”バッファオーバーフロー攻撃の対策手法の調査”, 石ヶ守祐, 平野学, 電気関係学会東海支部連合大会, 2006.

5. ”IC カードを用いたお薬手帳システムの開発”, 高橋功, 平野学, 電気関係学会東海支部連合大会, 2006.
6. ”公開鍵証明書に基づくスマートカードとリバースプロキシを用いた WWW サーバの選択手法”, 南谷真哉, 平野学, 電気関係学会東海支部連合大会, 2005.
7. ”スマートカードを利用した学内無線 LAN 環境における出席管理システムの設計と実装”, 三宅一正, 平野学, 電気関係学会東海支部連合大会, 2005.

A.5 Research Grant

1. 経済産業省, 未踏ソフトウェア創造事業（2007 年度第 I 期）
2. 経済産業省，未踏ソフトウェア創造事業（2005 年度下期）