Access Control for e-Business on MOPASS

Kilho Shin and Masahiro Taguchi

Abstract— ***MO(bile)PASS(port) is a consortium lead by Japanese major electric appliance manufacturers for the next-generation smart card. Its scope includes the software middleware that bridges between the smart card and real e-Businesses. In fact, the specification of Ticket Authentication Protocol, which provides access control functionality applicable to a wide range of e-Businesses (e.g. digital ticketing, digital content distribution), is under consideration by MOPASS. The specification is open and direct since it is based on a new methodology characterized by user's access-rights being straightforwardly authenticated by PKI. In addition, it covers the functionality of revoking, duplicating and transferring of access-rights in a peer-to-peer manner.***

Index Terms— ***Access control, authentication, e-Business, smart card, MOPASS, peer-to-peer, PKI***

I. Introduction

MOPASS (http://www.mopass.info) is a consortium for the promotion of the next-generation smart card named MOPASS Card. The Japanese major electric appliance manufacturers, Hitachi Ltd., Matsushita Electric Industrial Co., Ltd. and Toshiba Corp. are leading it and more than 60 companies from diverse fields participate in it. In this paper, by smart card we mean an IC card with a micro computer embedded therein.

Not merely MOPASS focuses on the hardware architecture of MOPASS Card but also it aims to pursue the possibility of MOPASS Card being widely used in e-Business.

Manuscript received June 30, 2004.

Kilho Shin is with Research Center of Advanced Science and Technology (RCAST), the University of Tokyo, Japan (e-mail: kshin@mpeg.rcast.u-tokyo.ac.jp).

Masahiro Taguchi is with Fuji Xerox Co., Ltd., Japan (e-mail: masahiro.taguchi@fujixerox.co.jp)

As is widely accepted, the smart card will definitely play more and more important roles in the future e-Business scenes due to its security capability to protect confidential information and vulnerable programs from adversarial accesses (e.g. theft, tampering and interruption). This requirement comes from the observation that e-Business practices cannot help involving crucial information being processed at the user's point. For example, when consumers order goods over the Internet, they are possibly required to prove their identity for payment based on their secrets (e.g. credit card number, PIN). If a program running on a client PC executes the calculations involving the secrets, they would be exposed to the threat of being stolen. In fact, the current architecture of PC only provides very weak means to prevent such adversarial behaviors of malicious programs.

The smart card has a potential to solve this problem and this is the reason why applying the smart card to e-Business is attracting a lot of attention of the industries. The framework of the smart card securing e-Business is very simple: a program confined within the smart card executes all the sensitive calculations and does not reveal anything but the safe results; a host program running on a PC receives the results and executes the consecutive calculations.

Based on this fundamental framework, MOPASS aims to define MOPASS Card as a security anchor for diverse e-Business services. Eventually, MOPASS not only will implement in MOPASS Card those security functions that are commonly useful for e-Business, but also it intends to open their API (Application Program Interfaces) to the public.

Currently, MOPASS has approved that the access control functionality that is commonly applicable to various e-Business

services is an important instance of the functionality to be implemented in MOPASS Card. This is because, as far as services are of the nature that only authorized users (e.g. the users who have paid for the services) are to be permitted to enjoy them, access control is commonly mandatory.

Nevertheless, current e-Business services tend to take the redundancy of deploying respective methods of access control. This redundancy causes the following disadvantages.

- The user and the provider of e-Business services have to take the overhead of supporting multiple methods (e.g. retaining plural smart cards).

- System developers are obliged to pay the redundant cost of developing different methods for different e-Business services.

- The exchangeability between e-Business services is harmed. For example, the content data currently tailored to respective content distribution systems are not mutually exchangeable.

Based on this understanding, the authors have proposed to MOPASS a protocol specification of access control, which supports the currently recognized requirements. The protocol is named Ticket Authentication Protocol (TAP) and its specification is given in [10].

By TAP, the user could use his or her single MOPASS Card as a *universal token* to access arbitrary services (Figure 1).

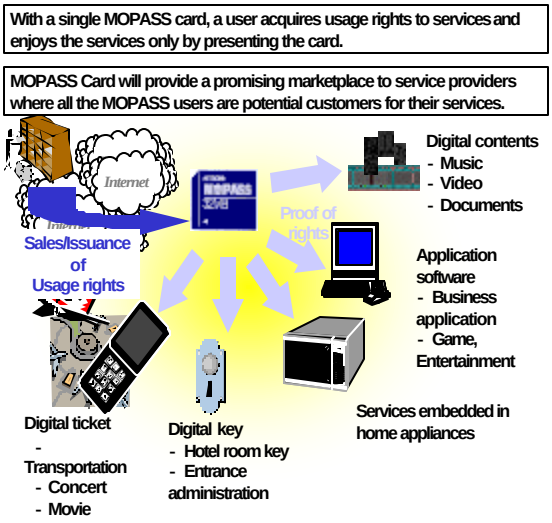

Figure 1. Aim of MOPASS Card

From a viewpoint of the service provider, they would be released from the burden of developing and distributing programs to be installed in smart cards, since all the necessary functions with respect to the access control functionality are provided through the open interface of MOPASS Card. Another important feature of TAP is to provide methods to revoke, consume, duplicate, modify and transfer an issued access-right in a peer-to-peer manner. For example, transferring an access-right is executed through communications between the two MOPASS Cards of the sender and recipient users. At the same time, the methods are secure in the sense that their execution never infringes the rules that the issuer of the access-right specified when it issued the access-right.

Currently, MOPASS has accepted TAP as a working draft, and a business trial is under preparation, which is going to involve real content and real consumers.

II. DESIGN PRINCIPLES

A. ARCHITECTUAL PRINCIPLES

The access control functionality of TAP is designed to support the following architectural principles.

1) Independence and Applicability

The access control functionality shall be totally separable from any other functionality (e.g. payment) and commonly applicable to a wide variety of services as far as they need it.

2) Support of PAC Model

PAC Model (3.A) introduces three independent players, namely Provider, Application and Card, and formalizes the processes of access control as communications between them.

- *Provider* denotes a provider of e-Business services.

- *Application* is a program, device, apparatus or their arbitrary combination to render e-Business services for the sake of their users.

- *Card* denotes MOPASS Card retained by a user of e-Business services. PAC Model tries to model the real world faithfully and consequently will make it possible to support a wide range of the real business models.

3) No Card Application Programs

In the conventional usage of smart card, it is common that service providers are required to develop card application programs for their own services and to persuade users to install the programs in their smart cards. This convention is, of course, a significant burden for both the service providers and the users.

4) Open Platform Design

Arbitrary systems shall be able to utilize the access control functionality of MOPASS Card through open interfaces. In particular, users' access-rights shall be authenticated only based on public information such as public keys of the public-key-cryptography.

Also, this principle means the stronger requirement that Provider, Application and Card shall cooperate with each other without knowing each other's secrets at all.

5) Non Server-Centric Design

Each phase of the access control functionality of TAP shall be completed through communications between two of Provider, Application and Card. For example, Provider and Card cooperate with each other in the phase of granting rights and no other party is involved. Transferring an access-right from a user to another is carried out by cooperation between two Cards of the sender and the recipient.

B. FUNCTIONAL PRINCIPLES

TAP also supports a plurality of functional requirements, some of which are shown below.

1) Enforcement of access rules

In practice, an access-right accompanies access rules (e.g. an expiry, permitted usage times). Application and Card, cooperating with each other, shall enforce the access rules when the right is executed.

2) Manipulation of access-rights after issuance

Manipulating the access-rights that were granted in the past may be important in practical business cases. In particular, revoking, consuming, duplicating, modifying and transferring access-rights are important, because their combination covers the most part of the required manipulations: *revoking* an access-right is to make it useless; *consuming* an access-right is to restrict its execution up to a specified number of times; *duplicating* an access-right is to make its duplicates; *modifying* an access-right is to change the access rules accompanying it; *transferring* an access-right is to transfer it from a user to another.

To prevent abuse of those manipulations, they shall not be executed beyond the permission specified when the access-right was initially issued .

3) Authentication of Application by Card

Since PAC Model defines that Provider and Application are mutually independent, Provider needs means to authenticate Application during Provider's proprietary service is being rendered. To avoid the overhead of communications between Provider and Application, Card shall authenticate Application on behalf of Provider.

III. UNDERLYING CONCEPTS

Two important concepts underlie TAP. One is PAC Model and the other is Digital Qualification.

A. PAC (Provider-Application-Card) MODEL

PAC Model is important because it defines chief players of the access control functionality of MOPASS Card and because it clarifies the scope of the functionality.

1) The roles of Provider, Application and Card

PAC Model gives the following roles to Provider, Application and Card, and further introduces *Ticket*, which are digital credentials representing permission for a particular user to render a particular service.

Provider grants a user an access-right to a particular service by issuing Ticket. Ticket is deliberately designed secure so that it can be delivered even by insecure means (e.g. http, SMTP).

Application inspects Ticket to verify that it properly represents the access-right of the relevant user. For this, Application receives a proof from Card that the user retains. Since the proof is generated from the secret proper to the Card, any different Card can never generate the same proof.

Card not only sends a proof to Application but also authenticates Application on behalf of Provider. An important feature of PAC Model resides in that Provider, Application and Card are defined to be totally independent of one another.

Provider being independent of the other two is necessary to model the real world faithfully and therefore to support a wide range of the real business models. For example, some electric appliance manufacturers are currently selling portable memory devices ([3]), which play the equivalent role of Card in their audio content download services: an audio player renders the audio contents only when they are stored in the memory device. While the manufacturers play the roles of the vender of Application and the issuer of Card, a record label, which is totally independent of the manufacturers, does the role of Provider.

In contrast, the independence between Application and Card is not necessarily to model the real world. In fact, the vender of Application and the issuer of Card in the previous example are the same manufacturers. However, this dependence is recognized as one of the most significant negative factors that are currently blocking spread of the smart card, since only governments, banks and credit card companies are currently recognized as the organizations that are capable of distributing smart cards over consumers at an inexpensive price.

Also, the multi-application operating systems of Java Card API and MULTOS ([2]) seem to support this requirement. With Java Card API and MULTOS, card application programs can be downloaded onto smart cards after the issuers issued them.

2) The scope of TAP

PAC Model assumes that Provider, the vender of Application and the issuer of Card are independent of one another. This assumption leads us to the conclusion that TAP shall rigidly specify the syntax, semantics and encoding rules of those messages to be exchanged among the players since they could not cooperate with each other otherwise.

Actually, the scope of TAP includes the following messages.

- A message to request Provider to issue Ticket.
- Ticket that Provider issues in response to the Ticket request.
- A sequence of messages exchanged between Application and Card for the mutual authentication.
- A message (`ApplDef`) that Provider generates to describe Application(s).

By contrast, although Application and Card cooperate with each other to render the services that are proprietary properties of Provider, PAC Model defines that TAP shall NOT specify any concrete means for Provider to establish trust with Application or Card. This is because means of establishing the trust is in practice affected by the actual business circumstances to which TAP is applied. For example, even if TAP specified that a single authority shall issue all the certificates to establish the trust, all the player could not accept the policies of the authority.

What TAP specifies regarding the trust is only that the consequence of establishing the trust must be Provider accepting the public keys of Application and Card. Provider should cope with those accepted public keys as follows.

- Provider specifies Applications' public keys in the data of `ApplDef` (5.A.1) and publicizes them.
- Provider executes key agreement protocols with Card using the public keys of Card (5.B).

B. DIGITAL QUALIFICATION

To support the principles of *Independence and Applicability* and *Open Platform Design*, TAP is based on a new methodology for access control, namely Digital Qualification ([11]). Digital Qualification applies the public-key-cryptographic techniques to access control in a straightforward manner.

1) Identifying a service

Provider generates a public key pair and assigns it to a service that Provider intends to provide to users. While the public key of the pair is publicized as a public identifier of the service, Provider shall store the private key secretly.

2) Granting an access-right

When a user requests an access-right to the service, Provider generates Ticket by transforming the private key of the service according using a certain cryptographic method. Ticket is issued to the requesting user.

3) Authenticating an access-right

Authenticating the user's access-right is carried out by inspecting Ticket. Application uses the public key assigned to the relevant service to inspect Ticket and executes calculations compliant to the corresponding public-key-algorithm.

Thus, Digital Qualification shows a clear contrast with the legacy of the initial idea of PKI (Digital Identification): the result of the execution of the public-key-algorithm in Digital Qualification is authentication of an access-right of the user, while it is authentication of the identity of the user in Digital Identification.

IV. Related Works

Applying PKI to access control itself is not new. The traditional method using ACL (Access Control List) is comprised of two steps: the first step to authenticate the user who is requesting access; the second step to look up ACL to inspect whether the user actually has the access-right. It is common that the first step is based on PKI. This ACL-based method is redundant in comparison with Digital Qualification, which achieves the same goal by a single step and doesn't require a user to reveal his or her identity.

In addition, looking up ACL requires extra processing time and network connection. In addition, maintaining a huge ACL is a troublesome task especially in terms of security, since modification of ACL would result in unauthorized access. Also, providers of nonfree services may not be interested in who the user is, but in whether the user has paid and not. Moreover, revealing the identity during authentication may be regarded as infringement of the privacy.

Plural companies currently deploy respective access control methods in their digital content distribution services ([4, 6]), which are commonly based on the symmetric-key cryptographic techniques. This approach improves the performance of execution but sacrifices the supporting of *Open Platform Design*. Furthermore, since the methods tend to be associated with particular contents formats, the design principle of *Independence and Applicability* is not supported.

V. Outlines of Specification of TAP

This section gives a brief illustration of the technical content of TAP.

A. Important Data Types

First of all, two important data type, `ApplDef` and `ApplRule`, are illustrated.

1) `ApplDef`

To prevent an abuse of services, Provider must draw a line between the trusted Applications that Provider permits to render its proprietary services and all the others.

The data type `ApplDef` is defined to support this requirement. Provider generates an instance of `ApplDef` and specifies the public keys of the trusted Applications in it. Further, Provider publicizes it and Card refers to it to authenticate Applications.

2) `ApplRule`

The data type `ApplRule` is to describe the access rules that Application and Card shall refer to in rendering the associated service. TAP specifies that every Ticket shall contain `ApplRule`.

`ApplRule` is comprised of the elements of `ClearPart` and `OpaquePart`. Card interprets and executes the access rules specified in `ClearPart`, while Provider specifies application-specific access rules in `OpaquePart`, which only particular Application can interpret.

B. Granting access-rights

On receipt of a request from Card, Provider generates Ticket.

For generating Ticket, Provider and Card agree on a one-time secret *k* in accordance with the unilateral version of Menezes-Qu-Vanstone Key Agreement (MQV-KA, [5, 7]). MQV-KA is a variant of Diffie-Hellman Key Agreement (DH-KA, [5, 8]) and satisfies the following properties.

- *k* is affected by the public key of Card.
- *k* is always random, and therefore a different secret is generated for a different Ticket.

To generate Ticket, Provider uses the secret in combination with `ApplDef`,

ApplRule and the public key *S* assigned to the requested service.

- The involvement of the random secret is to realize the revocation of Ticket.

- The involvement of ApplDef and ApplRule is to prevent them from being tampered with (7.B).

- The involvement of *S* is to prevent forging Ticket (7.A).

On the other hand, Card stores the secret in its secure storage and uses it to prove the user's access-right (5.C) against Application.

C. Authentication Protocol

Application and Card communicate with each other for respective purposes: Application aims to inspect Ticket to authenticate the user's access-right, while Card aims to authenticate Application.

As the base algorithms for Application to verify Ticket, TAP deploys two instances of the public-key algorithms: Schnorr Identification (SI, [9]) and DH-KA. The former is based on the zero-knowledge interactive proof techniques and is known very efficient, and therefore is suitable for the business cases where the efficiency is strongly required (e.g. digital ticketing). By contrast, the execution of the latter requires heavier calculations but it provides the functionality of decrypting secret data (e.g. a content key used in digital content distribution).

Since Ticket is generated from the secret *k*, ApplDef, ApplRule and the public key S, Card has to use those values to respond to Application.

On the other hand, authenticating Application is realized based on the MAC (message authentication codes) techniques. Application and Card share a MAC-generation key for a start and Card verifies MAC sent by Application. Since the key is derived from the public key of Application, which is specified in ApplDef, verifying MAC is identical with authenticating Application.

D. Revoking, consuming, duplicating and modifying Ticket

Given Ticket that Provider correctly generated, Card is able to revoke, consume, duplicate and modify the Ticket. Card executes the manipulations only when *privileged* Application instructs Card to do so. Whether or not Application is privileged is specified in ApplDef.

To revoke Ticket, Card deletes the one-time secret *k* from its secure storage: since Card can no longer respond to Application without *k* (5.C), Ticket becomes totally useless.

Consuming Ticket is used to limit the number of times of rendering services. The limit is specified as the value of the AvConsumeTimes element of ApplRule (5.A.2). The following are the steps that Card follows to consume Ticket.

1. Verify that Application is privileged to instruct Card to consume Ticket.
2. Verify that the value of AvConsumeTimes of ApplRule is positive.
3. Modify ApplRule so that AvConsumeTimes decreases by 1.
4. Generate a new one-time secret randomly.
5. Generate a new Ticket based on the new secret and the modified ApplRule.
6. Discard the initial secret *k* to revoke Ticket.

Thus, the initial Ticket is revoked and a new instance of Ticket is generated as a result of consuming Ticket.

Duplicating Ticket is executed following the steps similar to the steps for consuming Ticket, except that two new instances of Ticket are generated as a result.

Modifying Ticket is used to modify ApplRule arbitrarily. Privileged Application is responsible for specifying the modified ApplRule.

E. Transferring Ticket

Different from the other manipulations on Ticket, transferring Ticket is executed between two Cards. One is owned by the user to whom Ticket was initially issued and the other is owned by the user who will receive the transferred Ticket.

Briefly speaking, the recipient Card sends public keys to the sender Card and the sender Card returns the transferred Ticket. Since neither the public keys nor Ticket is secret data, the communication can be executed by any means including insecure protocols (e.g. http, SMTP) and exchange of physical storages (e.g. floppy disk).

The number of occurrence of transferring Ticket is limited by the value of the `AvTransferTimes` element of `ApplRule`, which Provider specified when it generated Ticket. The sender Card follows the steps stated below.

1. Execute MQV-KA with the recipient Card to share a new one-time secret. The public key of the recipient Card is used in the calculation.
2. Verify that the value of `AvTransferTimes` is positive.
3. Modify `ApplRule` so that `AvTransferTimes` decreases by 1.
4. Generate a new Ticket which involves the newly shared secret and modified `ApplRule`.
5. Discard *k* to revoke Ticket.

VI. SUPPORT OF DESIGN PRINCIPLES BY TAP

Since Section 5 shows all the functional principles stated in 2.B are supported, this section focuses on support of architectural principles.

Since TAP provides a specification for access control protocols at an abstract level, *Independence and Applicability* is apparently supported. TAP also specifies the syntax, semantics and encoding rules of the messages to be exchanged among Provider, Application and Card, and therefore supports *PAC Model*.

The protocols specified in TAP are to be executed between two of Provider, Application and Card. Especially, since neither server nor card application program is involved, *No Card Application Programs* and *Non Server-Centric Design* are supported. Moreover, TAP is public information and the secrets to be shared between players are all random and one-time. This means that *Open Platform Design* is also supported.

VII. CONSIDERATION OF SECURITY

A. Security against forging Ticket

The definition of Ticket is carefully designed so as to satisfy the strongest sense of the security: even if an adversary collected all the valid Tickets ever issued, the adversary could not extract any information useful to forge a new instance of Ticket. This property is provable assuming a secure pseudo-random-generation function $h(x,y)$, which is to be used in the definition of Ticket.

B. Security of the authentication protocol

The authentication protocol (5.C) is secure in the sense of the Bellare-Rogaway's definition ([1]). In more details, it satisfies the stronger sense of security (see also 7.C): if Application succeeds in verifying Ticket, it doesn't share the MAC-generation key with anyone but Card. It is also provable that Card reveals nothing to whoever impersonates Application.

C. Integrity of `ApplDef` and `ApplRule`

Tampering with `ApplDef` and `ApplRule` could be a serious threat to the authentication protocol, since it would allow impersonation of Application and abuse of services. However, the authentication protocol satisfies the property that, if tampered `ApplDef` or `ApplRule` is input to Application and/or Card, Application always fails in verifying Ticket. The following is a brief explanation of this property.

If either `ApplDef` or `ApplRule` input to Card is tampered with, Card no longer correctly responds to Application since the response from Card is calculated using both `ApplDef` and `ApplRule`.

If either `ApplDef` or `ApplRule` input to Application is different from that input to Card, Card fails to verify MAC since Application generates MAC using both of `ApplDef` and `ApplRule`.

D. Security of Revoking, Consuming, Modifying, Duplicating, and Transferring Ticket

Since tampering with `ApplDef` to impersonate privileged Application or tampering with `ApplRule` to abuse the manipulations makes Card return only an incorrect response, the adversary gains nothing (7.C). However, the adversary may attempt to make valid Tickets useless by impersonating privileged Application. Although Card never returns the valid response, it would be mislead to discard the one-time secret *k*. This attack is, of course, avoided by letting Card to verify Ticket in the same way as Application does.

VIII. CONCLUSION

The access control functionality that supports the design principles of *Independence and Applicability, PAC Model, No Card Application Programs, Open Platform Design* and *Non Server-Centric Design* will definitely provide smart card with the potential to be used as a *universal token*: a user could access arbitrary services only by presenting his or her single smart card to prove his or her access rights.

The authors have developed a specification of access control protocols that supports all the principles. Moreover, the specification satisfies security requirements at a very high level: most of security achievements are proved based on a mathematical model. The specification is proposed to MOPASS, which is a consortium for the next generation smart card.

REFERENCES

[1] Bellare, M. and Rogaway, P., "Entity authentication and key distribution". *In Lecture Notes in Computer Science – Advances in Cryptology – Crypto '93,* Vol. 773, 1994, pp. 232 – 249

[2] MAOSCO Ltd., "MULTOS Getting Started", MAOSCO.Ltd., 1999

[3] 4C Entity, LLC., "Content Protection for Recordable Media Specification, SD Memory Card Book", *4C Entity*, 2002

[4] 4C Entity, LLC., "Content Protection for Recordable Media (CPRM) Specification. Version, Introduction and Common Cryptographic Elements, Revision 1.0", *4C Entity,* 2003

[5] IEEE, "P1363-2000: Standard Specifications for Public Key Cryptography", 2000

[6] Keitaide-Music Consortium, "Keitaide-Music Technical Specification Version 0.9", http://www.keitaide-music.orc/spec/index_e.html, 2000

[7] Menezes, M., Qu, M. and Vanstone, S., "Some new key agreement protocols providing implicit authentication", *2nd Workshop on Selected Areas in Cryptography (SAC '95),* Ottawa, Canada, 1995, pp.18 – 19

[8] RSA Laboratory, "PKCS #3: Diffie-Hellman Key Agreement Standard", *RSA Laboratories Technical Note*, 1993

[9] Schnorr, C.P., "Efficient identification and signatures for smart cards", *In Lecture Notes in Computer Science – Advances in Cryptology – Crypto 89,* Vol. 435, 1990, pp. 239-252

[10] Shin, K. and Taguchi, M., "Ticket Authentication Protocol for MOPASS", *a technical document of MOPASS (in Japanese)*, 2003

[11] Shin, K., "Digital Qualification: An Approach To Infrastructures Of Access Control for Internet Commerce". *SSGRR 2001*, L'Aquila, Italy, 2001