Practical Anonymous Access Control Protocols for Ubiquitous Computing

Kilho Shin and Hiroshi Yasuda
Research Center for Advanced Science and Technology, The University of Tokyo
Email: {kshin, yasuda}@mpeg.rcast.u-tokyo.ac.jp

***Abstract*— Privacy has been a central concern of ubiquitous (pervasive) computing. The boundary between private and public moves dynamically depending on the context in which the issue is considered. As for access control for ubiquitous computing, the concept of *consensual disclosure* is an answer to the question of where the boundary should be drawn — Unless a user gives their explicit consent to opening the identity, perfect privacy, that is, anonymity and unlinkability are supported. This paper presents concrete protocols for anonymous and unlinkable access control that are appropriate for ubiquitous computing partly in the sense that they support consensual disclosure. The protocols presented in this paper are practical because they are efficient and based on a model that represents the real world. In particular, they support a characteristic of the key transfer: a service appliance acquires keys to decrypt contents of services, if, and only if, it accepts proof of access rights as presented by a user, and can prove that it is trusted by the service provider. Moreover, the protocols are provably secure. This study has been sponsored by the Ministry of Economy, Trade and Industry, Japan (METI) under contract, New-generation Information Security R&D Program.**

I. INTRODUCTION

In a ubiquitous (pervasive) computing environment, users can enjoy access to various kinds of services and resources without restrictions on time and location. From the viewpoint of protecting services and resources, this also implies that the security functionalities of trust management, authentication and access control are the mandatory building blocks for ubiquitous computing.

However, naive implementation of these security functionalities can certainly invoke an invasion of privacy. Since networked sensors for ubiquitous computing easily collect information about users' personal identities and access histories, a malicious entity, which could be an organization, a system administrator or an application manufacturer, may possibly use the collected information in ways such that the users would never permit. It has been reported that this problem of privacy has been of central concern to people [1], [2].

It is also widely accepted that ubiquitous computing requires security measures that are more flexible than those for accessing the Internet. In terms of privacy, Palen and Dourich give the following description [3].

> *Privacy management is not about setting rules and enforcing them; rather, it is the continual management of boundaries between different spheres of action and degrees of disclosure within those spheres. Boundaries move dynamically as the context changes. ... The significance of information technology in this view lies in its ability to disrupt or destabilize the regulation of boundaries.*

Certainly, the boundary between private and public is dynamic, since we "disclose or publicize information about ourselves, our opinions and our activities, as means of declaring allegiance or even of differentiating ourselves from others" [3]. Thus, perfect privacy is far from ideal.

Where should the boundary between private and public for ubiquitous computing be drawn? Our answer to this problem is the concept of *consensual disclosure* — if, and only if, users give their explicit consent, their identity is revealed only during the relevant access event. In addition, their identity remains anonymous for any other access events, whether they have already happened in the past or will happen in the future.

In one aspect, this paper aims to present a set of practical access control protocols that support anonymity and unlinkability based on *consensual disclosure*.

On the other hand, trust management is not an issue no less important than privacy in ubiquitous computing.

Blaze et al. [4], [5] advocate a *distributed trust management model*, which presents a methodology to express and handle the varying trust structures of ubiquitous computing, and in particular, to allow users to arbitrarily migrate from one domain to another where different trust policies control access to services and resources. This model is also important because it is a natural extension of the trust management models widely accepted for the access to the Internet (e.g. X.509, SPKI).

One of the central concepts of the distributed trust management model is known as *authorization*. Here, the term "authorization" is used to represent the binding of entity authentication and access control into a unified process. For example, a basic SPKI certificate associates a public key pair with not only the identity of its subject but also the roles, qualifications, privileges and other attributes assigned to the subject. Hence, verifying that an entity possesses the private key of the key pair indicates that the relevant entity is the one identified by the certificate, and also assigns to the entity the roles, qualifications or privileges specified in the certificate.

The access control scheme presented in this paper develops the idea of authorization so that a public key pair is defined as being separate from an individual entity and represents roles, qualifications or privileges assigned to the entity. This feature contributes to realizing anonymous access control and also to making our access control scheme conform to the distributed trust model at a more flexible level.

Thirdly, ubiquitous computing is becoming more applicable in the real world fueled by the recent development of infrastructures, such as the Internet, short-range wireless communication (e.g. Bluetooth, IEEE 802.11) and cellular phones. This also implies that technologies for ubiquitous computing take the complexity and the heterogeneity of the real world into consideration.

In this paper, we describe a trust model that represents the heterogeneous trust structures of the real world. The access control protocols in this paper are based on this trust model.

II. Related Work

A. Distributed Trust Management

The term of *distributed trust management* was first introduced by Blaze et al. [4], [5]. Distributed trust management must meet the following two requirements:

- combining entity authentication and access control into a unified process (authorization);
- incorporating local policies and delegation of access rights into the existing trust management methodology based on PKI.

By introducing an authorization certificate, Simple Public Key Infrastructure (SPKI, [6]) partially supports the above requirements and can be regarded as the first step towards distributed trust management. A basic SPKI certificate defines the binding between authorization and a public key pair. Therefore, verifying possession of the private key of the key pair immediately indicates that the certificate's subject has access rights to certain services or resources as specified in the certificate.

Moreover, the certificate's subject can issue a new certificate meaning that the authorization of the original certificate is to be delegated to the subject of the new certificate (whether or not the delegation itself is permitted may be specified in the original certificate).

The methodology of the authorization certificate for SPKI can take the place of the existing approach of centralized and nested ACLs (Access Control Lists), and enable a distributed management of authorization.

However, to make trust management flexible to such a level so that it can be applicable to ubiquitous computing, supporting authorization and delegation is yet insufficient. In fact, there exist a number of administrative entities in a distributed system of ubiquitous computing and each of these entities may have different trust policies. In other words, proofs of trust (e.g. a chain of authorization and delegation certificates for SPKI) may be understood differently, depending on the administrative entity.

Blaze et al. [4], [5] present an elaborate and formal scheme to incorporate local policies into the existing trust management and to fill the gap between differing understanding. Some systems [7], [8] are known to be designed based on this scheme, in which a service agent, who receives a request for access to the services or resources for which it is responsible, submits a *query* to the *PolicyMaker* system. On receipt of the query, *PolicyMaker* returns proof, if present, that determines whether the requested action can be permitted to the requesting user. Queries to *PolicyMaker* take the following form:

$$\mathit{key}_1, \mathit{key}_2, \ldots, \mathit{key}_n \quad \text{REQUESTS} \quad \mathit{ActionString}$$

The *key*s are the public keys of which the requesting user claims to possess the private counterparts, and *ActionString* indicates the action requested by the user. *PolicyMaker* makes up the answer to the query of one or more *assertions*, each of which is of the following form:

$$\mathit{Source}\ \text{ASSERTS}\ \mathit{AuthorityStruct}\ \text{WHERE}\ \mathit{Filter}$$

- *Source* indicates the relevant local policy if the assertion is a policy assertion, and specifies a public key if the assertion is a signed assertion. A policy assertion is axiomatic, and hence, is unconditionally evaluated as being valid. In contrast, a signed assertion is accepted, if, and only if, the public key specified as the *Source* is trusted and the associated signature is successfully verified using the public key.
- *AuthorityStruct* specifies the *structure* of one or more public keys. It may specify a single public key, a set of public keys or a more complicated structured list of public keys (e.g. any k instances of the n public keys in the list).
- *Filter* specifies a predicate that returns either true or false upon an input of the *ActionString*.

If a signed assertion includes a single public key as *AuthorityStruct* and the valid predicate, which returns the value *true* irrespective of input, as *Filter*, the assertion is semantically identical to an ordinary X.509 certificate.

Upon receipt of a query, *PolicyMaker* generates and returns the smallest acyclic directed graph that satisfies the following: each vertex of the graph is labeled by a single assertion; any starting vertex of the graph, which is not the target vertex of any arc, is a policy assertion; for each arc, the assertion of its source vertex asserts *Source* of the assertion of its target vertex; and the assertions corresponding to the terminal vertices, any of which is not the source vertex of any arc, collectively assert the public keys specified in the query. The key holder of the query is recognized as being qualified for the action requested in the query, if, and only if, the signatures of all the signed assertions appearing in the graph are successfully verified and *ActionString* satisfies all *Filters* specified in the assertions.

The *PolicyMaker* scheme, however, does not specify concrete ways of how a service agent verifies that the requesting user actually possesses the *key*s that the user claims to possess. This may be due to a plurality of algorithms associated with the scheme and a lack of known and explicit requirements.

However, for ubiquitous computing, it is crucial that users agree on a method to verify their access rights with service agents from any location. Of course, the ideal way to support this requirement is to standardize a unique method. In the worst case, a negotiation mechanism may be required through which a device worn by a user and a service agent can agree on such a method from several candidates.

The importance of this requirement is made clear, by considering the following scenario.

> *Alice always wears a smart badge that contains her private keys. One day, she visited a building where her friend Bob works. As soon as she entered the building, she was faced with problems: she could not access the directory service, she was not allowed to get off the elevator at the floor where Bob's office was located, and she could not call Bob using an internal phone. Since she knew that Bob had delegated the authorization necessary for Alice prior to her visit, she could not understand why there were such problems. The delegation was performed by a standardized means of delegation credentials. Afterward, she realized the cause behind the problems — her smart badge was not compatible with the service agents in Bob's building.*

One of the focuses of this paper is, based on Blaze's distributed trust management scheme, to propose a concrete protocol by which a service agent communicates with users' devices to verify their access rights. For this purpose, the requirements that such protocols should support, particularly in a ubiquitous computing scenario, are clarified. For example, anonymity and unlinkability are two of the recognized requirements.

B. Group signatures

The following features characterize group signatures.

1) A group signature is a digital signature that a member of a group produces on behalf of the group.
2) Although whoever has an access to a predetermined group verification (public) key can verify the group signature, the identity of the individual who generated the group signature is never revealed.
3) Only a trusted group authority (TGA) has an ability to identify the individual member who generated the group signature. The TGA may perform this function when there are disputes, etc.

Chaum and Heijst [9] first introduced the concept of group signatures, and since then, various technical proposals to realize group signatures have been made [9]–[12].

As an application of group signatures, Chen [11] proposed a secure electronic auction system that excludes malicious participants from participating while protecting the privacy of honest participants through using a bid document, which is signed by a bidder and is always anonymously verifiable. If a dispute arises, the auction organizer works as the TGA, and reveals the identities of malicious bidders, based on the signatures that they have produced.

Commonly, a group signature scheme consists of the SETUP, JOIN, SIGN, VERIFY and OPEN operations.

- *SETUP*: The TGA generates a group public key pair. The TGA publicizes the public component of the pair for verification of group signatures, and withholds the private component.
- *JOIN*: In cooperation with the TGA, a member of the group generates a group signing key for themselves.
- *SIGN*: A member of the group signs an arbitrary message using the group public key and their own group signing key.
- *VERIFY*: A verifier uses only the group public key to verify signatures that members of the group generate.
- *OPEN*: If a dispute arises, the TGA identifies the individual who signed a particular message. For this purpose, the TGA uses the group public key pair.

A group signature scheme is required to support the following requirements.

- *Correctness*: A signature generated by a group member is accepted.
- *Unforgeability*: A signature generated by anyone other than a member of the group is denied.
- *Anonymity*: In verifying the signature, only the TGA is able to identify the originator of a signature in verifying the signature.
- *Unlinkability*: Anyone other than the TGA is not able to correctly answer the question of whether two independent signatures were produced by the same member.
- *Exculpability*: The TGA and/or any group members who conspire with one another is/are not able to forge the signatures of any other group member.
- *Traceability*: The TGA is able to identify the originator of any given signature.

It is also desirable that a group signature scheme supports the requirement that the addition or withdrawal of members or a change in the group signing key of a member does not necessitate changing the group public key. A group signature scheme that supports this additional requirement is considered *dynamic*.

Camenisch and Stadler presented the first dynamic group signature scheme [10]. Their scheme has the advantage that the sizes of a group public key and generated signatures are independent of the size of a group.

The scheme that Ateniese, Camenisch, Joye and Tsudik present [13] has practical efficiency, in addition to all of the features stated above.

The following outlines the scheme by Ateniese et al [13].

1) Let QR_n be the group of the quadratic residues of a safe RSA composite number n. A composite $n = pq$ is said to be safe, if, and only if, $p =$

$2p' + 1$ and $q = 2q' + 1$ hold for some primes p', q'. Further, the TGA selects $a, d, g, g_1 \in_R QR_n$ and $x \in_R \mathbb{Z}^*_{p'q'}$, and calculates $y = g^x \bmod n$. The tuple (n, a, d, g, g_1, y) is publicized as a group public key, while the tuple (p, q, x) is preserved by the TGA as a group secret key.

2) A user U takes the following procedures to join the group.
   a) U and the TGA cooperate with each other to select x_u so that U and the TGA are respectively confident that x_U distributes uniformly in its domain. x_u is a secret of U and $y_u = a^{x_u} \bmod n$ is kept by the TGA.
   b) The TGA randomly selects a prime e_U and calculates c_u as shown below:
   $$c_u = (y_u d)^{\frac{1}{e_u}} \bmod n$$
   The strong RSA hypothesis guarantees that only the TGA, who knows p and q, can calculate (c_u, e_u).
   c) The TGA issues (c_u, e_u) to U. Eventually, (x_u, c_u, e_u) becomes a group signing key of U.
3) A group signature to a message m is non-interactive proof of the knowledge of (c_u, e_u, x_u) satisfying
   $$c_u^{e_u} = da^{x_u} \bmod n.$$
   According to the Fiat-Shamir heuristic, the challenge is dependent on m.
   In addition, U encrypts c_u with the key y, and issues the encrypted c_u with non-interactive proof, showing that the encryption is in the correct form.
4) For OPEN, the TGA decrypts c_u to reveal the identity of U.

More group signature schemes [14]–[16] have been presented after [13]. However, if we measure efficiency of verification by the number of times of executing scalar multiplication over an elliptic curve and modular exponentiation over a large composite or prime number, none of them is more efficient than the scheme by [13].

C. Anonymous credentials

Camenisch et al [17] present a highly sophisticated algorithmic scheme for anonymous credentials. More specifically, the scheme supports the following features.

- An authority can issue non-transferable credentials under the pseudonym of a user.
- A verifier can verify the fact that a user retains a credential issued by a certain authority without knowing his/her pseudonym or the credential.
- An authority can verify that a user, who is known to the authority by a certain pseudonym, retains a credential issued by a different authority.

The technical principles of the scheme are as follows.

1) *QR*$_n$ is the group of quadratic residues modulo an RSA composite n. The composite n is public information, while the prime factors of n are withheld by an authority. In addition, a, b and d are public elements of *QR*$_n$.
2) A pseudonym p of a user is an element of *QR*$_n$ such that
   $$p \equiv a^\alpha b^\beta \bmod n,$$
   where α and β are withheld by the user.
3) A credential that the authority generates is its *digital signature* to the pseudonym p. Precisely, the credential is a pair (c, γ) such that $c \in QR_n$ and
   $$c^\gamma \equiv pd \equiv a^\alpha b^\beta d \bmod n$$
   Consequently, γ is data such that only the authority can generate, while α and β are user secrets.
4) The user presents zero-knowledge proof of the fact that they know α, β and γ satisfying $c^\gamma \equiv a^\alpha b^\beta d \bmod n$. The proof does not reveal anything but a, b, c, d and n.

Idemix [18] is implementation of this scheme.

The problem of this scheme is the overhead of performing time-consuming zero-knowledge proof procedures. In particular, certain services require continual verification of credentials and the overhead of verification would harm the quality of the services. For example, a rendering service using video signals may be required to render contents only while the authorized persons are in front of the screen. To support this requirement, the service must continually verify the credentials of people in front of the screen and the resulting overhead would reduce the CPU capability that the service can allocate to decoding video signals.

In fact, sacrificing security, [19] presents a simple technique that does not involve cryptography to avoid such overhead.

Thus, another focus of this paper is to realize anonymous access control that requires only small amount of calculation to verify access rights.

In particular, when continual verification is required, the burden of executing the subsequent verification should be much lighter than that of executing the initial one. An example of a technique to support this requirement is to use the common key cryptography for the subsequent verification based on keys exchanged during the initial verification. Hence, the access control protocol presented in this paper is, in one aspect, a key exchange protocol.

D. Problems

Both group signatures and anonymous credentials seem to be applicable to the anonymous access control required for ubiquitous computing. However, they, in fact, involve some deficiencies to be used for this purpose.

First, *traceability* for group signatures requires that THE TGA is capable of *open*ing the identity of the originator of a signature. This means that in accessing services and resources, the privacy of users is always threatened by an overpowering "big brother", namely the TGA, who is authorized to open users' identities without their consents at all. In particular, we should note that the role of the TGA is played by a service provider or a resource manager, who determines whether access rights are to be granted to users. Since the service provider and the resource manager are potential *enemies* to users, against whom users want to protect their privacy the most, this setting contradicts one of the fundamental requirements for anonymous access control.

On the other hand, it is also a fact that access to certain services and resources requires having users to reveal their identities. For example, entering a critical area, manipulating an important machine or browsing confidential information may require recording the identities of accessed users.

An answer to this problem may be the concept of *consensual disclosure*, which represents that, if, and only if, a user gives their explicit consent, their identity is revealed during verification of their access rights. In addition, any other anonymous access events performed by this user remain anonymous and unlinkable.

Secondly, the schemes known to be the most efficient for group signatures and anonymous credentials may not be efficient enough to be applied to access control for ubiquitous computing. For example, both the group signature scheme by Ateniese et al. [13] and the anonymous credential scheme by Camenisch et al. [17] requires 22-time execution of the modular exponentiation for the single occurrence of simple verification of a signature or possession of a credential. In ubiquitous computing, it is naturally presumed that access control events occur much more frequently, and therefore the requirement for efficiency in ubiquitous computing must be stricter. Thus, while the efficiency of the schemes presented in [13]–[16] and [17] may be sufficient for group signatures and credentials, they may not be applicable to access control for ubiquitous computing.

In particular, when a service is continuously provided during a certain period of time, the efficiency of *continual authentication* of access rights is important. In the previous scenario of a rendering service using video signals, it is desirable that continual and subsequent authentication be executed more efficiently than the initial authentication of access rights.

III. UNDERLYING MODEL

A. Players

The design for the access control protocol presented in this paper is based on a model that represents the real world. The model defines four players: *User Agent* (*UA*), *Service Provider* (*SP*) *Service Provider Agent* (*SP-A*), and *Service Appliance* (*SA*).

1) SP-A*:* In ubiquitous computing, it is unrealistic to assume that users' accesses are controlled in a centralized manner (*e.g.* by means of an ACL (Access Control List) in a centralized server), since access may occur in an *off-line* environment. Moreover, since a user migrates from one domain to another at will, a user naturally carries their access rights (*e.g.* credentials) with them. In this setting, unauthorized transfer of access rights by malicious users are certainly a serious threat to the security of services and resources.

We present two basic measures to counter the threat: prevention and after-the-fact detection. In the real world, prevention is more desirable than after-the-fact detection, since even a single occurrence of unauthorized access to critical resources may be fatal.

Two approaches exist for prevention: one approach assumes safe (tamper-resistant) hardware at the user's point, while the other does not. Camenisch et al. [17] propose a strong measure for the latter, namely *all-or-nothing* non-transferability, where a user transfer of a single instance of credentials results in a transfer of all the credentials that the user retains. However, the proposed scheme for all-or-nothing non-transferability [17] is not efficient enough in practice. Moreover, it is not certain that the measure is effective in the real world.

Thus, this paper assumes the existence of a safe hardware module in the computer that a user carries with them. However, this setting may pose the threat of a *big brother*; from the user's view, the hardware, a black box, may leak private information. To avoid this threat, we deploy the idea of the *wallet-with-observer* model [20]: a black box hardware module (an observer) supervises the user's behaviors, while communication between the observer and the outside is under full control of the user. Precisely, the *SP-A* play the role of an observer.

From a practical point of view, the tamper-resistance characteristic of the *SP-A* is the key to maintaining security in the scheme, and in particular, to preventing unauthorized transfer of access rights (*e.g.* credentials). Recent applications like UIM (User Identification Module) for cellular phones and digital cash schemes that deploy smart card technology have proven that this tamper-resistance technology is effective.

Thus, the *SP-A* has the following properties:

- The *SP-A* works on behalf of *SP*. In other words, *SP-A* prevents abuse of access rights including unauthorized transfer of access rights.
- The *SP-A* always works as a "slave": it is basically inactive. It wakes up when an outside program accesses it through the equipped interfaces and remains awake until the access ceases.
- The *SP-A* is tamper-resistant. The programs and data that exist within the *SP-A* are protected from any sort of access from the outside.

2) UA*:* The *UA* is a module that exists in the computer that a user carries with them, and is controlled by the user. The *UA* plays the role of supervising the *SP-A* on behalf of the user so that Token will not leak any data that possibly infringe on anonymity or unlinkability.

The following is an example of the implementation of *UA* and *SP-A*. The *SP-A* is a smart chip embedded in a cellular phone, and the *UA* is the device driver program installed in the same cellular phone that interfaces the *SP-A* with the OS of the cellular phone.

3) SP*:* The *SP* is an operator of services and is the only entity eligible to grant users access rights to the services.

In granting access rights, the *SP* communicates with the *SP-A* via the *UA*.

4) SA*:* The *SA* is a software program, device, or apparatus that renders services on behalf of the *SP*. The *SA* communicates with the *SP-A* via the *UA* to verify access rights granted to the user carrying the *SP-A*.

B. Trust between players

To represent the real world faithfully, it is extremely important that the four players are defined to be completely independent of each other. In other words, any two of them independently determine whether to trust each other based solely on mutual agreement.

For example, assume that the *SP* is a record label (*e.g.* Sony BMG Music Enterprise), and provides digital contents of music to portable music players (*e.g.* iPod), which play the role of the *SA*. In this setting, the manufacturers of portable music players (*e.g.* Apple Corp.) are companies independent of the *SP*. This indicates that the *SP* cannot control the *SA* in any sense. As another example, when the *SP-A* is incorporated into a cellular phone, it is likely that the *SP-A* is manufactured by the same manufacturer as the cellular phone (*e.g.* Motorola), which is different from the *SP* or the manufacturers of the *SA*. Thus, to represent the real world, the four players are assumed to be independent of each other.

In addition, it is necessary to understand that trust between the players is arbitrary and heterogeneous. A record label may provide its contents only to portable players manufactured by major manufacturers (*e.g.* Apple Corp, Panasonic), and may reject small third party ventures for security reasons. Conversely, it is natural that a single music player may play back music provided by different *SP*'s (*e.g.* Sony BMG Music Entertainment, Toshiba EMI), and refuse pirated contents.

To support the complexity and heterogeneity of trust between the players, the following is defined by the model.

- The *SP* permits the *SA*'s to render its services, if, and only if, it trusts them and they are authenticated at the point of access events. We call this feature *verifier authentication*
- Reversely, the *SA* renders services of those the *SP*'s that it trusts.
- The *SP* grants access rights to its services to only the users who carry the trusted *SP-A*.
- In contrast, the *SP-A* does not have any preference for particular *SP*'s. It is the user that draws a line between the trusted and non-trusted *SP*'s.
- The *SA* by itself does not have any preference for a particular *SP-A*, either. It renders services, if, and only if, the following conditions are satisfied: the *SP-A* can present a valid proof of possession of access rights, and the *SP-A* authenticates the *SA* on behalf of the *SP* for the relevant services.
- The *SP-A* by itself does not have any preference for a particular *SA*, either. The *SP-A* only inspects whether the *SA* that it is communicating with is trusted by the *SP*.

IV. REQUIREMENTS

Based on the discussion in the previous sections, the access control scheme presented in the subsequent sections is designed to support the following requirements.

- Anonymity and unlinkability.
- Compliance to the distributed trust management model presented by Blaze et al. [4], [5].
- Verifier authentication.
- Efficiency.
- Provable security.

The following is a small, detailed illustration on "anonymity and unlinkability".

To protect services from unauthorized access, the *SP-A* is required to play the following roles on behalf of the *SP*:

- The *SP-A* prevents users from unauthorized transfer of granted access rights;
- The *SP-A* cooperates with the *SA* to prevent users without valid access rights from accessing services.
- The *SP-A* prevents unauthorized *SA* from rendering proprietary services of the *SP*s.

Thus, the *SP-A* which remains in a user's computer, is regarded as an agent for the *SP*. This implies that the *SP-A* may work against the user, and, in particular, may leak the identity of the user to the *SP* and the *SA*.

The role of the *UA* is to prevent the *SP-A* from infringing on the user's privacy by interfacing the *SP-A* with the *SP* and the *SA*. Therefore, two interfaces exist: one between the *UA* and the *SP*/*SA*, another between the *UA* and the *SP-A*.

Thus, the requirement for anonymity and unlinkability is refined as follows.

a) Interface between the UA *and the* SP/SA*:* The *UA* randomizes (anonymize) the messages that are to be issued to the *SP* or the *SA*.

b) Interface between the UA *and the* SP-A*:* There are no means to prevent the *SP-A* from submitting arbitrary messages. Therefore, if the *SP-A* colludes with an adversary who can observe communication through this interface, the adversary may acquire information, thus invading the user's privacy. The best possible measure to this invasion is the assurance that, if the *SP-A* deviates from the specification of the protocol, the *UA* will detect it.

V. BASIC PROTOCOLS

In this section, the basic protocols for granting and verifying access rights are presented.

For the key transfer protocol, refer to Section VI.

A. Outline of protocols

One of the basic ideas in achieving the goal of anonymity and unlinkability is to use a public key pair to identify a service [21]. Since verifying users' access rights is carried out using the assigned public key instead of a user's certificate, in principle, the *SA* does not use information that may reveal users' identities.

Our access control scheme is comprised of the following three phases.

1) Identifying services: The *SP* generates a public key pair and assigns it to a service that it intends to provide to users. The public key of the pair (service public key) is publicized as a public identifier of the service, while the *SP* withholds the private key (service private key).

2) Granting access rights: By request from a user, the *SP* mathematically transforms the requested service private key and issues the transformed data, or *Access ID*, to the requesting user. This is a one-way transformation with a trap door; it is impossible to reveal the service private key from the Access ID without the key that the *SP* and the user's *SP-A* exchanged prior to the transformation.

3) Verifying access rights: The *SA* verifies that the *anonymized* Access ID presented by a user was properly generated from the private key of the requested service. Since the verification of the anonymized Access ID is performed using only the public key of the service, and the *UA* randomizes the messages that the *SP-A* generates, the communication does not reveal any information indicating the user's identity or linking the current access to any past or future accesses by the same user.

B. Notation

From here, this paper employs the following notation.

- $\mathcal{G}_T$ and $\mathcal{G}_S$ are both additive groups. $\mathcal{G}_T$ ($\mathcal{G}_S$) is associated with the the *SP-A* (*Service*, resp.) and is used for granting (verifying, resp.) access rights.
- The notation $x \overset{\mathcal{G}_T}{=} y$ ($x \overset{\mathcal{G}_S}{=} y$) indicates that x and y are identical to each other in $\mathcal{G}_T$ ($\mathcal{G}_S$, resp.).
- $\mathcal{G}_T$ ($\mathcal{G}_S$) satisfies the property that solving the equation $A \overset{\mathcal{G}_T}{=} xB$ ($A \overset{\mathcal{G}_S}{=} xB$, resp.) in x is intractable for randomly selected A and B from $\mathcal{G}_T$ ($\mathcal{G}_S$, resp.).
- G_T (G_S) is a base element in $\mathcal{G}_T$ ($\mathcal{G}_S$, resp.) with orders n_T (n_S, resp.).
- $T \overset{\mathcal{G}_T}{=} \tau G_T$ is a public key of the *SP-A*, while τ is its private counterpart, which is securely confined within the *SP-A*. An *SP-A* group shares the same (T, τ).
- $S \overset{\mathcal{G}_S}{=} \sigma G_S$ is a public key of the service, while σ is its private counterpart, which is securely confined within the *SP*.
- $A \overset{\mathcal{G}_S}{=} \alpha G_S$ is a public key of the *SA*, while α is its private counterpart, which is securely confined within the *SA*.
- $\pi(x)$ is a one-to-one mapping that transforms an element x of $\mathcal{G}_T$ or $\mathcal{G}_S$ into a bit string of a fixed length.
- $\omega(x)$ is a pseudo random function, which takes a variably long bit string as an input and outputs a bit string of a fixed length. Since $\omega(x)$ is a pseudo random function, it is also one-way and collision-free.
- $\mu(key, x)$ is a secure MAC generation (verification) function.

C. Access ID

The Access ID is data that the *SP* issues to a user as a representation of the access rights granted to the user. The Access ID, denoted by *aid*, is generated through cooperation between the *SP* and the *SP-A*. In fact, *aid* is the service private key σ of the relevant service masked with a random secret k shared between the *SP* and the *SP-A*.

$$\textit{aid} = \sigma - k \bmod n_S.$$

The *UA* may anonymize the Access ID when requested to present it to the *SA*. The anonymized form of the Access ID, denoted by *anm*, is the Access ID *aid* masked with a random secret $\rho \in [0, n_S)$ generated by the *UA*.

$$\textit{anm} = \textit{aid} - \rho \bmod n_S$$

D. Anonymous rights granting protocol

To generate and issue an Access ID, the *SP* has to share a random secret k with the *SP-A*. The following are the requirements that the protocols used for sharing k must support.

- The *SP* authenticates an *SP-A* group without knowing the individual *SP-A* that initiated the communication for sharing k.
- The *SP* does not share k with anyone but the initiating *SP-A*. In particular, any other *SP-A*, even if it belongs to the same group, does not know k.
- The *UA* randomizes the messages that the *SP-A* submits to the *SP* to prevent the user's identity from leaking from the *SP-A*.

Figure 1 depicts the protocol for granting rights, which is based on the *unilateral* version of the MQV (Menezes-Qu-Vanstone) protocol for exchanging keys [22].

The public key T represents an *SP-A* group instead of an individual instance. In other words, all of the *SP-A*s belonging to the same group share the same private key τ. An *SP-A* group may consist of products that are accredited by an authority or manufactured by a trusted manufacturer. Because of these settings for T, the *SP* cannot distinguish between instances of the *SP-A*.

On the other hand, only the instance of the *SP-A* that knows ϵ_T and ϵ_U can actually calculate k. Therefore, an attack, such as those in which malicious *SP-A*s generate common ϵ_T to share k is not effective, since the *UA* prevents ϵ_U from leaking. Moreover, the *UA* randomizes E_T by using a random ϵ_U, and therefore the *SP* cannot obtain any information about a user's identity even if the *SP* and *SP-A* collude with each other.

Authentication of the *SP* is assumed to be carried out in an out-of-band manner. A technical reason for this is because it is not desirable to assume that the *SP* always has its public key in $\mathcal{G}_T$. Moreover, from a practical point of view, it is natural to assume that the rights granting protocol works over the existing authentication protocols

SP		*UA*		*SP-A*
			$\xleftarrow{E_T}$	$\epsilon_T \in_R [0, n_T)$ $E_T \overset{\mathcal{G}_T}{=} \epsilon_T G_T.$
		$\epsilon_U \in_R [0, n_T)$ $E_U \overset{\mathcal{G}_T}{=} \epsilon_U G_T + E_T$		
$\epsilon_P \in_R [0, n_T)$ $E_P \overset{\mathcal{G}_T}{=} \epsilon_P G_T$ $k = \pi(\epsilon_P(E_U + \bar{E}_U T))$ $\mathit{aid} = \sigma - k \bmod n$	$\xleftarrow{E_U}$			
	$\xrightarrow{\mathit{aid}}$	Store *aid*.		
	$\xrightarrow{E_P,\#}$	$\longrightarrow$	$\xrightarrow{E_P,\epsilon_U}$	$E_U \overset{\mathcal{G}_S}{=} (\epsilon_U + \epsilon_T)G_S$ $k = \pi((\epsilon_U + \epsilon_T + \bar{E}_U \tau)E_P)$ Store k.

Figure 1. Rights granting protocol

(*e.g.* SSL), whose deployment may be required by outside applications.

Anonymity of the rights granting protocol is apparent, since what *SP* gains through the communication is only E_U, which uniformly distributes over $\mathcal{G}_T$ due to $\epsilon_U \in_R [0, n_T)$.

E. Anonymous rights verifying protocol

The *SA* communicates with the *SP-A* through the *UA* to verify the access rights of the user carrying the *SP-A*. In the same communication, the *SP-A* examines, on behalf of *SP*, whether the relevant *SA* is one of the *SA*s trusted by the *SP*.

Figure 2 depicts the anonymous version of the rights verifying protocol, which supports the following requirements.

- The *SA* examines an anonymized Access ID to verify whether the *SP-A* retains the secret k.
- The *UA* randomizes all of the messages that the *SP-A* submits to the *SA* in order to hide the identity of the user.
- The *UA* can detect if the *SP-A* deviates from the specification of the protocol.

The protocol is designed based on the Schnorr identification algorithm [23], which fulfills the condition for perfect ZKIP (Zero-Knowledge Interactive Proof). The difference between our scheme from the original Schnorr scheme is that the *SP* selects c from a much wider interval $[0, n_S)$. This modification reduces the probability of a dishonest *UA* cheating the *SA* so that it is almost nonexistent, even if the *SA* and the *UA* exchange the triplet (W, c, r) of a witness, challenge and response only once. In exchange, this modification harms the simulatability of the scheme in that the simulator may not stop in polynomial time. To fix this deficiency, our scheme makes the *SP* submit the cryptographic secure one-way hash $\omega(c)$ of the challenge c prior to receiving the witness W. By this, our scheme becomes a statistical ZKIP.

The following are to be noted regarding the anonymous rights verifying protocol.

- The *SA* verifies r received from the *UA* by:
$$(r + c \cdot \mathit{anm})G_S \overset{\mathcal{G}_S}{=} cS + W \qquad (1)$$
- The *UA* verifies r' received from the *SP-A* by:
$$(r' + c \cdot \mathit{aid})G_S \overset{\mathcal{G}_S}{=} cS + W' \qquad (2)$$
- By verifying Eq. (2), the *UA* can investigate whether the *SP-A* deviated from the specification of the protocol.
- If the *UA* successfully verifies Eq. (2), an attacker who can observe the communication between the *UA* and the *SP-A* can only perform the *guess-and-verify* attack, where the attacker has a list of one or more guesses of *aid*'s and tests whether each guess makes Eq. (2) hold.
- The *SP-A* verifies that Service Appliance is trusted by the *SP* (verifier authentication). For this purpose, the *SA* and the *SP-A* first exchange a secret using the Diffie-Hellman key exchange algorithm and then respectively derive a MAC key (*i.e. key*) from the exchanged secret. The *SP-A* authenticates the *SA*by verifying the MAC generated by the *SA* using the *key*,

F. Proof of anonymity and unlinkability for V-E

The following discussion asserts the anonymity and unlinkability of the communication between the *SA* and the *UA*.

We consider the following two cases.

If the *UA* successfully verifies Eq. (2), then the communication between the *UA* and the *SA* is a statistical ZKIP. Therefore, what the *SA* learns from the communication is only *anm*, but as *anm* uniformly distributes over $[0, n_S)$, this implies that the *SA* learns nothing after all.

If *UA* fails to verify Eq. (2), what *SA* gains through the communication are only *anm* and W. Since the two independent random numbers ρ and w'' uniformly distribute over $[0, n_S)$, *anm* and W are also independent of each other and uniformly distribute over their respective domains.

SA	*UA*	*SP-A*
	$\rho \in_R [0, n_S)$	
	$\mathit{anm} = \mathit{aid} - \rho \bmod n_S$	
$c \in_R [0, n_S)$	$\xleftarrow{anm}$	
$e_0 = \omega(c)$	$\xrightarrow{e_0}$ $\longrightarrow$ $\xrightarrow{e_0}$	$w' \in_R [0, n_S)$
	$\xleftarrow{W'}$	$W' \overset{\mathcal{G}_S}{=} w'G_S$
	$w'' \in_R [0, n_S)$	
	$W \overset{\mathcal{G}_S}{=} w''G_S + W'$	
$\mathit{key} = \omega(\pi(\alpha W) \vert c)$	$\xleftarrow{W}$	
$e_1 = \omega(\mathit{key})$	$\xrightarrow{c, e_1}$	
	$\xrightarrow{c, e_1, w''}$	$\mathit{key} = \omega(\pi((w' + w'')A) \vert c)$
		Unless e_0 and e_1 are valid, abort the session.
	$\xleftarrow{r'}$	$r' = ck + w' + w'' \bmod n_S$
	Unless r' is valid, abort the session.	
	$r = r' + c\rho \bmod n_S$	
Unless r is valid, abort the session.	$\xleftarrow{r}$	

Figure 2. Anonymous rights verifying protocol

G. Non-anonymous rights verifying protocol

Figure 3 depicts the non-anonymous protocol for verifying rights. The important differences between this non-anonymous version and the anonymous version are as follows.

- The *UA* presents *aid* instead of *anm* to the *SA*.
- The *SP-A* signs r'.

H. Proof of non-anonymity and unlinkability for V-G

In terms of non-anonymity and unlinkability, the following are the requirements that the non-anonymous protocol for verifying rights must support.

- The *SA* can detect if the *UA* presented *anm* instead of *aid*.
- What *SA* learns from the communication is *aid* and (U, s).

The following discussion asserts that the protocol given by Figure 3 supports the above requirements.

The *UA* can modify the signature s so that it can be verified using *anm* instead of *aid*. The modification is performed only by adding $\omega(\pi(U) \mid r')\rho$ to s, where $\rho \equiv \mathit{aid} - \mathit{anm} \bmod n_S$. In fact, the following equality holds.

$$(s + \omega(\pi(U) \mid r')\rho + \omega(\pi(U) \mid r')\mathit{anm})\, G_S \\ \overset{\mathcal{G}_S}{=} \omega(\pi(U) \mid r')S + U$$

However, the *UA* cannot modify r' in the same way as it can do s, since r' is signed by *SP-A*. Hence, the *UA* can cheat the *SA* if, and only if, it has obtained a signature to $r' + c\rho$ by the *SP-A* in the precedent conversations with *SP-A*, since the Schnorr signature scheme is non-malleable. However, this is unlikely to happen, since the *UA* has to present *anm* prior to receiving r'. At the moment that *UA* presents *anm*, the probability that the *UA* receives c and r' such that a signature to $r' + c\rho$ exists in the list of the received signatures is vanishingly small. This immediately asserts that the non-anonymity of our protocol.

Supporting of the second requirement of unlinkability is due to the fact that the protocol is a statistical ZKIP, if we neglect (U, s) sent in the last message to the *UA*.

VI. Key transfer

A. Protocol specification

The key transfer is the feature that the *SA* receives a secret key that the *SP* has generated, if, and only if, the *SA* succeeds in the rights verifying protocol. That is, the *SA* verifies the possession of k by the *SP-A*, and the *SP-A* authenticates the *SA* based on the *SA*'s public key A. Apparently, this feature augments the support for verifier authentication.

Let K denote the key that the *SP* intends to transfer to the *SA*. K satisfies $K \overset{\mathcal{G}_S}{=} \kappa S$, where $\kappa \in_R [0, n_S)$ is a random number that the *SP* generates. For the transfer, the *SP* issues $L \overset{\mathcal{G}_S}{=} \kappa G_S$ to the *SA*, and the *SA* recaptures K through the protocol.

The key transfer protocol consists of an anonymous version and a non-anonymous version, as is the same for the rights verifying protocol. Figure 4 depicts the anonymous version.

In performing the protocol, the *SA* does not send raw L, since any observer of the communication including *UA* could also reveal K, otherwise.

1) The *SA* selects a random $\lambda \in [0, n_S)$, and then calculates $C \overset{\mathcal{G}_S}{=} \lambda L$.
2) The *SA* sends C instead of L to the *SP-A* via the *UA*.
3) On receipt of R from the *UA*, the *SA* calculates the following.

$$\begin{aligned} K' &\overset{\mathcal{G}_S}{=} \mathit{anm} \cdot C + R \\ &\overset{\mathcal{G}_S}{=} (\sigma - k - \rho)C + (k + \rho)C \\ &\overset{\mathcal{G}_S}{=} \sigma C \overset{\mathcal{G}_S}{=} \lambda\kappa S \end{aligned}$$

SA		*UA*		*SP-A*
	$\xleftarrow{aid}$			
$c \in_R [0, n_S)$				
$e_0 = \omega(c)$	$\xrightarrow{e_0}$	$\longrightarrow$	$\xrightarrow{e_0}$	$w', u' \in_R [0, n_S)$
			$\xleftarrow{W', U'}$	$W' \overset{\mathcal{G}_S}{=} w'G_S \quad U' \overset{\mathcal{G}_u{}'}{=} G_S$
		$w'', u'' \in_R [0, n_S)$ $W \overset{\mathcal{G}_S}{=} w''G_S + W'$		
$key = \omega(\pi(\alpha W)\|c)$	$\xleftarrow{W}$			
$e_1 = \omega(key)$	$\xrightarrow{c, e_1}$			
			$\xrightarrow{c, e_1, w'', u''}$	$key = \omega(\pi((w' + w'')A)\|c)$ Unless e_0 and e_1 are valid, abort the session. $r' = ck + w' + w'' \bmod n_S$ $U \overset{\mathcal{G}_S}{=} U' + u''G_S \bmod n_S$
			$\xleftarrow{r', s}$	$s = \omega(\pi(U) \mid r') \cdot k + u' + u'' \bmod n_S$
		$U \overset{\mathcal{G}_S}{=} u''G_S + U'$ Unless r', U, s are valid, abort the session. $r = r'$		
Unless r, U, s are valid, abort the session.	$\xleftarrow{r, U, s}$			

Figure 3. Non-nonymous rights verifying protocol

SA		*UA*		*SP-A*
		$\rho \in_R [0, n_S)$ $anm = aid - \rho \bmod n_S$		
$c \in_R [0, n_S)$	$\xleftarrow{anm}$			
$e_0 = \omega(c)$	$\xrightarrow{e_0}$	$\longrightarrow$	$\xrightarrow{e_0}$	$w' \in_R [0, n_S)$
			$\xleftarrow{W'}$	$W' \overset{\mathcal{G}_S}{=} w'G_S$
		$w'' \in_R [0, n_S)$ $W \overset{\mathcal{G}_S}{=} w''G_S + W'$		
$key = \omega(\pi(\alpha W)\|c)$	$\xleftarrow{W}$			
$e_1 = \omega(key)$	$\xrightarrow{c, C, e_1}$			
		$\underline{x, y \in [0, n_S)}$ $\underline{U \overset{\mathcal{G}_S}{=} xG_S + yC}$		
			$\xrightarrow{c, e_1, w'', C, U}$	$key = \omega(\pi((w' + w'')A)\|c)$ Unless e_0 and e_1 are valid, abort the session. $\underline{R' \overset{\mathcal{G}_S}{=} kC}$ $\underline{V \overset{\mathcal{G}_S}{=} kU}$
			$\xleftarrow{r', R', V}$	$r' = ck + w' + w'' \bmod n_S$
		Unless r' and R' are valid, abort the session. $r = r' + c\rho \bmod n_S$ $\underline{R \overset{\mathcal{G}_S}{=} R' + \rho C}$		
Unless r is valid, abort the session. $\underline{K' \overset{\mathcal{G}_S}{=} anm \cdot C + R}$	$\xleftarrow{r, R}$			

Figure 4. Anonymous key transfer protocol

4) Finally, the *SA* recaptures $K \overset{\mathcal{G}_S}{=} (\lambda^{-1} \bmod n_S)K'$.

A typical example of the usage of K is to use it as keys that encrypt contents of the relevant service. For example, let *SA* be a music player that provides users with a service of playing back music. From the viewpoint of the *SP* (*e.g.*a record label), the clear (decrypted) music data shall be processed only by the *trusted* music players, which are only those music players (*i.e.* the *SA*) that the *SP-A* authenticates on behalf of the *SP*. This requirement is supported using the key transfer feature, since only the trusted *SA* is capable of recapturing K.

B. Proof of anonymity and unlinkability

To supports anonymity and unlinkability, the *UA* has to verify that R' is the right form, or that R' is identical with kC. Otherwise, the *SP-A* could send arbitrary data revealing the identity of the user. For this purpose, we deploys a well known method for investigating a decisional

Diffie-Hellman tuple — a method that determines whether $z \equiv xy \bmod n$ for a given quadruple (G, xG, yG, zG), where G is an arbitrary point on an elliptic curve and n is the order of G.

In fact, on receipt of R' from the *SP-A*, the *UA* investigates whether $(G_S, S - \textit{aid} \cdot G_S, C, R')$ is a decisional Diffie-Hellman quadruple, or in other words, whether R' is identical to kC without knowing k. For this purpose, the *UA* performs the following.

1) The *UA* selects a pair of mutually independent random numbers $x, y \in [0, n_S)$.
2) The *UA* calculates $U \overset{\mathcal{G}_S}{=} xG_S + yC$ and submits U to the *SP-A*.
3) On receipt of R' and V from the *SP-A*, the *UA* verifies Eq. (3).

$$V \overset{\mathcal{G}_S}{=} x(S - \textit{aid} \cdot G_S) + yR' \tag{3}$$

If the *SP-A* calculates R' and V correctly by $R' \overset{\mathcal{G}_S}{=} kC$ and $V \overset{\mathcal{G}_S}{=} kU$, it is apparent that the *UA* has successfully verified Eq. (3).

Conversely, if Eq. (3) holds, $R' \overset{\mathcal{G}_S}{=} kC$ holds except for an almost nonexistent, small probability. This fact is proven as follows. For a fixed U, there exist n_S pairs of (x, y) satisfying $U \overset{\mathcal{G}_S}{=} xG_S + yC$, since

$$\log_{G_S} U = x + \lambda\kappa y \bmod n_S. \tag{4}$$

On the other hand, Eq. (3) implies that

$$\log_{G_S} V \equiv xk + y \log_{G_S} R' \bmod n_S. \tag{5}$$

Therefore, unless $\log_{G_S} R' \equiv \lambda\kappa k \bmod n_S$, only a single (x, y) satisfies Eq. (5) out of the n_S candidates that satisfy Eq. (4). Since all what the *SP-A* can know about (x, y) is that it satisfies Eq. (4), the probability that the *UA* succeeds in verifying Eq. (3) with an incorrect R' is only $\frac{1}{n_S}$, which is vanishingly small.

The deployment of this method allows the key transfer protocol to support anonymity and unlinkability.

If the *UA* successfully verifies Eq. (3), the SA gains no new knowledge from the communication except for the value of σC. In other words, since generating R is *simulatable* with the knowledge of *anm* and σC (*i.e.* σL), the protocol remains a statistical ZKIP. This means that the communication is anonymous and unlinkable.

What should be noted here is that L does not contain any information indicating the identity of the user. For example, consider the scenario where L is contained in the header of encrypted data of a music file. If a user downloads the music file from the same site where they purchased *aid*, the site (*i.e.* the *SP*) has the opportunity to generate L so that it indicates the identity of the user. Thus, to protect his/her own privacy, the user should acquire music file data through procedures completely separate from the procedures used for purchasing *aid*.

VII. Supporting of the Requirements

The previous sections have already showed that out of the requirements presented in Section IV, our access control scheme supports "anonymity and unlinkability", "verifier authentication", and "provable security".

A. Efficiency

First, readers should note the following.

- The efficiency of the protocol is impacted the most by the calculation performed by the *SP-A*, since the computing power of the *SP-A* is usually restricted.
- Scalar multiplication over an elliptic curve and equivalently modular exponentiation are much heavier than any other calculation including addition over an elliptic curve, a hash function, MAC generation and verification, and modular addition and multiplication.

In this regard, our rights verifying protocol based on the Schnorr zero-knowledge identification algorithm [23] is efficient, since the *SP-A* has to execute scalar multiplication over $\mathcal{G}_S$ only 2 times for the witness W and the shared secret *key*. In addition, the calculation of witnesses can be done prior to execution of the protocol, if the *SP-A* has enough room in its memory to store pre-calculated witnesses.

As for the total number of times of executing scalar multiplication and the modular exponentiation, our scheme requires 6, while the schemes by Ateniese et al. [13] and Camenisch et al. [17] require 22.

In the situation where continual verification between the *SA* and the *SP-A* is required, verification is performed very efficiently using the shared secret *key*.

B. Compliance to the distributed trust management model

It is easy to extend the protocol so that verification by *SA* is based on multiple service public keys and multiple *aid*. This extension is necessary to fully conform to the Blaze's distributed trust management model [4], [5].

Let (S_i, σ_i) be the public key of the i-th service ($i = 1, \ldots, k$). Then, the i-th anonymized Access-ID *anm*$_i$ is defined by

$$\textit{anm}_i = \sigma_i - k_i - \rho_i \bmod n_S.$$

For the challenge c, *SP-A*calculates the response r by

$$r = \sum_{i=1}^{k} \omega(c|i)(k_i + \rho_i) + w \bmod G_S.$$

Therefore, the *SA* verifies r by

$$\left(\sum_{i=1}^{k} \omega(c|i)\textit{anm}_i + r\right) G_S \overset{\mathcal{G}_S}{=} \sum_{i=1}^{k} \omega(c|i) S_i + W.$$

Kilho Shin received the B.S. and M.S. from the University of Tokyo, Japan in 1982 and 1984 respectively. Then, he joined the Corporate Research Laboratory of Fuji Xerox, Co., Ltd. After he left Fuji Xerox, he has belonged to Research Center for Advanced Science & Technology (RCAST), the University of Tokyo, as a project researcher since 2004.

Hiroshi Yasuda received the B.E., M.E. and Dr.E. from the University of Tokyo, Japan in 1967, 1969, and 1972 respectively. Then, he had joined the Electrical Communication Laboratories of NTT in 1972. After served twenty-five years (1972-1997), with the last position of Vice President, Director of NTT Information and Communication Systems Laboratories at Yokosuka, he left NTT and has joined The University of Tokyo. He acted Director of The Center for Collaborative Research (CCR) for 2 years(2003-2005), and he is now a professor in CCR and Research Center for Advanced Science & Technology(RCAST). His study area is applied information technology. He has been involved in works on Video Coding, Image Processing, Tele-presence, B-ISDN Network and Services, Internet and Computer Communication Applications. Now he has started researches on DRM (Digital Rights Management), Network Security and ”Kansei” (more human) communication. He is now advocating collaboration between Industries and Academia. In International Standardizing Area, he had served as the Chairman of ISO/IEC JTC1/SC29 (JPEG/MPEG Standardization) from 1991 to 1999. He had also served as the President of DAVIC (Digital Audio Video Council) from September 1996 to September 1998. He received 1987 Takayanagi Award, 1995 the Achievement Award of EICEJ, 1995-1996 The EMMY from The National Academy of Television Arts and Science , 2000 Charles Proteus Steinmetz Award from IEEE , 2005 Takayanagi Award and also Ericsson Telecommunication Award 2006. He is Fellow of IEEE, EICEJ, and IPSJ, a member of Television Institute. He wrote ”International Standardization of Multimedia Coding” in 1991, ”MPEG/International Standardization of Multimedia Coding” in 1994, ”The Base for the Digital Image Coding” in 1995, ”The Text for Internet” in 1996, ”The Text for MPEG” in 2002 and ”The Text for Content Distribution” in 2003.

REFERENCES

[1] L. Barkhuus and A. Dey, “Location-based services for mobile telephony: a study of users’ privacy concerns,” in *INTERACT 2003, 9th IFIP TC13 International Conference on Human-Computer Interface*, 2003.

[2] E. Kaasinen, “User needs for location-aware mobile services,” *Personal Ubiquitous Comput.*, vol. 7, no. 1, pp. 70–79, 2003.

[3] L. Palen and P. Dourish, “Unpacking ”privacy” for a networked world,” in *CHI2003*, 2003.

[4] B. Blaze, J. Feigenbaum, and J. Lacy, “Decentralized trust management,” *IEEE Proceedings of the 17th Symposium*, 1996.

[5] B. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis, “The role of trust management ini distributed sistems,” *Secure Internet Programming, LNCS*, vol. 1603, pp. 185–210, 1999.

[6] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen, “SPKI certificate theory,” IETF, RFC 2693, September 1999.

[7] L. Kagal, V. Korolev, S. Avancha, A. Joshi, T. Finin, and Y. Yesha, “Centaurus: an infrastructure for service management in ubiquitous computing environments,” *Wirel. Netw.*, vol. 8, no. 6, pp. 619–635, 2002.

[8] L. Kagal, S. Cost, T. Finin, and Y. Peng, “A framework for distributed trust management,” in *IJCAI-01 Workshop on Autonomy, Delegation and Control*, 2001.

[9] D.Chaum and E. van Heist, “Group signatures,” in *In Advances in Cryptology — EUROCRYPT ’91, LNCS 547*, 1991, pp. 257–265.

[10] J. L. Camenisch and M. A. Stadler, “Efficient group signature schemes for large groups,” *Lecture Notes in Computer Science*, vol. 1294, pp. 410+, 1997. [Online]. Available: citeseer.ist.psu.edu/article/camenisch97efficient.html

[11] L. Chen and T. P. Pedersen, “New group signature schemes,” in *In Advances in Cryptology — EUROCRYPT ’94, LNCS 550*, 1995, pp. 171–181.

[12] D. X. Song, “Practical forward secure group signature schemes,” in *ACM Conference on Computer and Communications Security*, 2001, pp. 225–234. [Online]. Available: citeseer.ist.psu.edu/457688.html

[13] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik, “A practical and provably secure coalition-resistant group signature scheme,” *Lecture Notes in Computer Science*, vol. 1880, pp. 255–??, 2000. [Online]. Available: citeseer.ist.psu.edu/ateniese00practical.html

[14] J. Camenisch and A. Lysyanskaya, “Dynamic accumulators and application to efficient revocation of anonymous credentials,” 2002. [Online]. Available: citeseer.ist.psu.edu/article/camenisch02dynamic.html

[15] J. Camenisch and J. Groth, “Group signatures: better efficiency and new theoretical aspects,” 2004. [Online]. Available: citeseer.ist.psu.edu/article/camenisch05group.html

[16] J. Furukawa and H. Imai, “An efficient group signature scheme from bilinear maps,” in *ACISP 2005*, 2005, pp. 455–467.

[17] J. Camenisch and A. Lysyanskaya, “Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation,” in *EUROCRYPTO 2001, LNCS 2045*. Springer-Verleg, 2001, pp. 93–118.

[18] J. Camenisch and E. V. Herreweghen, “Design and implementation of the idemix anonymous credential system,” in *CCS ’02: Proceedings of the 9th ACM conference on Computer and communications security*. New York, NY, USA: ACM Press, 2002, pp. 21–30.

[19] G. Iachello and G. Abowd, “A token-based access control mechanism for automated capture and access systems in ubiquitous computing,” Georgia Institute of Technology, GVU Center, GIT Technical Report GIT-GVU-05-06, June 2005.

[20] D.Chaum and T.Pedersen, “Wallet databases with observers,” vol. 740, pp. 89–105, 1993.

[21] K. Shin, “Digital Qualification: An approach to infrastructures of access control for internet commerce,” in *SSGRR 2001*, 2001.

[22] M. Menezes, M. Qu, and S. Vanstone, “Some new key agreement protocols providing implicit authentication,” in *2nd Workshop on Selected Areas in Cryptography (SAC ’95*, 1995.

[23] C. Schnorr, “Efficient identification and signatures for smart cards,” in *Crypto ’89, LNCS 435*, 1990, pp. 239–252.