2006－UBI－12（10）
2006／11／10

Consensual Disclosure を実現する実用的な追跡不能アクセス制御方式

保黒　政大[†]　申　吉浩[††]　前田　勝之[†]　安田　浩[††]

[††] 東京大学 先端科学技術研究センター　〒153–8904 東京都目黒区駒場 4 丁目 6 番 1 号 先端研 3 号館 3 階
[†] 株式会社ディー・ディー・エス　〒450–0003 愛知県名古屋市中村区名駅南 1-27-2 日本生命笹島ビル 16F
E-mail: †{hoguro,maeda}@dds.co.jp, ††{kshin,yasuda}@mpeg.rcast.u-tokyo.ac.jp

あらまし　ユビキタスコンピューティングにおいて，プライバシーは大きな懸案事項である．ユビキタスコンピューティングの基本は，ユーザが携帯しているコンピュータデバイスと環境に偏在するコンピュータとのシームレスで透過的なやりとりである．これらのやりとりによりユーザは，いつでも，どこでも有用なサービスを享受することができる．しかし，ネットワーク接続された膨大な数のセンサがユーザの個人情報を収集できるとすれば，ユビキタスコンピューティングは管理社会を実現するための危険な手段となり，権力者は個人のあらゆる行動を監視するようになるであろう．追跡不能なアクセス制御はこのような危険を回避するための重要な鍵である．追跡不能性とは，ユーザアクセスの匿名性だけでなく，単一ユーザによる独立した複数のアクセスの事実を不可視にする必要がある．はじめに，ユビキタスコンピューティングにおける追跡不能なアクセス制御の必要条件を明確にする．従来より，匿名性と追跡不能性については暗号の観点から研究が進められている (例えば，グループ署名や匿名証明)．しかし，これらは全ての必要条件を満足しているわけではない．そこで，全ての必要要件を満たした実用的な追跡不能アクセス制御プロトコルを提案する．本研究は，経済産業省，新世代情報セキュリティ研究開発事業の研究として行われたものである．

Consensual Disclosure in practically unlinkable access control for ubiquitous computing

Masahiro HOGURO[†], Kilho SHIN[††], Katsuyuki MAEDA[†], and Hiroshi YASUDA[††]

[††] Research Center for Advanced Science and Technology, The University of Tokyo　3rd Bldg., 4–6–1 Komaba, Meguro-ku, Tokyo, 153–8904 Japan
[†] DDS Inc.　Nihonseimei Sasashima Bldg. 16F, 1-27-2 Meieki Minami, Nakamura-ku, Nagoya-shi, Aichi, 450–0003 Japan
E-mail: †{hoguro,maeda}@dds.co.jp, ††{kshin,yasuda}@mpeg.rcast.u-tokyo.ac.jp

Abstract　Privacy has been a central concern of the ubiquitous computing. The essential of the ubiquitous computing exists in seamless and transparent transactions between computing devices carried by users and computers ubiquitously existing in the environment. Through the transactions, users are provided with useful services anytime, anywhere. However, if a huge number of networked sensors in the environment could collect users' personal information, the ubiquitous computing would be dangerous leverage to realize a controlled society, where authorities would be capable of censoring every small activity of people. Unlikable access control is a critical key to avoid this danger: unlinkability requires not only anonymity of accesses by users but also hiding of the fact that two independent access events were performed by a single user. The first contribution of this paper is clarification of requirements for the unlinkable access control applicable to the ubiquitous computing. Although anonymity and unlinkability have been intensively studied in the cryptographic context (*e.g.*group signatures, anonymous credentials), those ever presented schemes turn out not to support all the requirements presented here. As the second contribution, we present a sketch of a practical scheme of unlikable access control, which support all the proposed requirements. A prototype of the scheme is planned to be implemented and tested with support from Ministry of Economy, Trade and Industry, Japan.

1. Introduction

Ubiquitous (pervasive) computing is approaching reality fueled by the recent development of infrastructures including the Internet, short-range wireless communication (e.g. Bluetooth, IEEE 802.11) and cellular phones. In a ubiquitous (pervasive) computing environment, users enjoy access to various kinds of services and resources anytime, anywhere. This also implies that technologies for seamless, transparent and secure access control are critical building blocks of ubiquitous computing.

However, naive implementation of access control will certainly invoke the problem of privacy invasion. Networked sensors of ubiquitous computing easily collect users' personal identifying information and their histories of access, so a malicious entity, which could be an organization, a system administrator or an application manufacturer, may possibly use the collected information in such ways that the users would never accept. Eventually, it is reported that the privacy problem has been a central concern of people [1], [2].

With respect to privacy, Palen and Dourich described as follows in [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. (snip) The significance of information technology in this view lies in its ability to disrupt or destabilize the regulation of boundaries.*

Apparently, the boundary between privacy and publicity 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]. At the same time, for protecting services and resources accessing individuals may be required to reveal their identities. For example, an authority may mandate that the information who accessed confidential information is logged.

Thus, *absolute* privacy protection is far from an ideal goal. So, what is the best balance between privacy and publicity for ubiquitous computing? Our answer is *consensual open*:

- the identity of an accessing user is revealed, if, only if, the user is requested and consent to open her identity; by default, her access is kept unlinkable.
- the user can deny the request at the sacrifice of the intended access;
- even when the user's identity is revealed at an occurrence of access, the other access events remain unlinkable.

The requirement of consensual open is different from that of *traceability* required for group signature, where a trusted group authority is capable of revealing the signer of a given signature at anytime at will.

Also, ubiquitous computing adds complexity to the setting for access control. For example, the premise that the entity that grants access rights and the entity that verifies them share a mutual interest is not necessarily true for ubiquitous computation, since the granting entity and the verifying entity don't always belong to the same domain. This implies that verification of users' access rights is not sufficient and authentication of the verifying entity is also necessary. We call this requirement *verifier authentication*.

Thus, access control for ubiquitous computing definitely has its own proper requirements including consensual open and verifier authentication. One of the chief contributions of this paper is having identified such requirements (Section 3. 1). In addition, in Section 4., a scheme of access control protocols supporting the identified requirements is presented.

2. Related work on anonymity and unlinkability

2. 1 Group signatures

The following features characterize group signatures.

（1） A group signature is a digital signature that a member belonging to 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, verifying the signature never reveals the individual member who generated it.

（3） Only a trusted group authority (TGA) has an ability to identify the individual member who generated the group signature. TGA may perform this function in case of dispute and so forth.

Chaum and Heijst [4] first introduced the concept of group signatures, and various technical proposals to realize group signatures have been made [4]〜[7].

The requirements that a scheme of group signatures shall support are well identified.

Correctness　A signature generated by a group member shall be accepted.

Unforgeability　A signature generated by anyone other than the members of a group shall be denied.

Anonymity　Anyone other than TGA shall not be able to identify the originator of a signature in verifying the signature.

Unlinkability　Anyone other than TGA shall not be able to answer the question whether two independent signatures were produced by a single member.

Exculpability　TGA and/or any group members, even if they collude with one another, shall not be able to forge signatures of any other group member.

Traceability　TGA shall be able to identify the originator of any given signature.

It is also desirable that a group signature scheme supports the requirement that join or withdrawal of members or change of a group signing key of a member does not require change of the group public key. A group signature scheme that supports this additional requirement is called *dynamic*.

The first dynamic group signature scheme was the one that Camenisch and Stadler presented [5]. Their scheme also has the advantage that the size of a group public key and that of generated signatures are independent of the size of a group. Also, the scheme that Ateniese, Camenisch, Joye and Tsudik presented [8] has practical efficiency in addition to all the features stated above.

2. 2 Anonymous credentials

Camenisch et al [9] presented a practical scheme of anonymous credentials, which is highly sophisticated. More specifically, the scheme supports the following features.

- An authority can issue non-transferable credentials under a pseudonym of a user.
- A verifier can verify the fact that a user retains a credential issued by a certain authority without knowing his 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 secrets of an authority. In addition, a, b and d are public elements of QR_n.

（2） A pseudonym p of a user is the element of QR_n such that

$$p \equiv a^\alpha b^\beta \bmod n,$$

where α and β are secrets of 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 a zero-knowledge proof of the fact that it knows 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 [10] is an implementation of this scheme.

2. 3 Problems

At a glance, both group signatures and anonymous credentials are applicable to unlinkable access control for ubiquitous computing. However, they, in fact, involve some deficiencies to be used for this purpose.

First, as previously stated, traceability of group signatures is inappropriate for ubiquitous computing, since privacy of users is always threatened by a mighty big brother, namely TGA. In particular, we should note that, since TGA would grant access rights to users, it is a potential enemies against whom users like to protect their privacy the most.

Secondly, the schemes known the most efficient for group signatures and anonymous credentials may not be efficient enough to be effectively applied to access control for ubiquitous computing. Both the group signature scheme by Ateniese et al. [8] and the anonymous credential scheme Camenisch et al. [9] requires 22-time execution of the modular exponentiation for a single occurrence of anonymous verification. Since it is naturally presumed that authentication of access rights would be performed much more frequently in ubiquitous computing, the efficiency of the schemes of [8], [9] may not be sufficient.

In particular, it is strongly desirable that *continual authentication* subsequent to the initial authentication of access rights is executed more efficiently. For example, a video rendering service may be required to render contents only while authorized persons are in front of the screen. To support the requirement, the service must continually perform verification during rendering, and the resulted overhead would reduce the CPU capability that the service can allocate to decoding video signals.

3. Requirements for ubiquitous access control

3. 1 Requirements recognized so far

The discussion so far makes us recognize the following requirements.

Unlinkability　It is impossible to determine whether two independent access events were performed by a single user. This logically implies anonymity of access events. In addition, unlinkability between events of granting access rights and a consequent access is also required.

Consensual open　The identity of an accessing user is revealed, if, and only if, the user gives her explicit consent to the reveal.

Verifier authentication　Only authorized verifier can execute verification of users' access rights and consequent operations including rendering of services.

Efficiency　Authentication of users' access rights can be executed efficiently enough assuming that the number of times of accesses would drastically increase in ubiquitous computing. In particular, it is desirable that continual authentication is much more efficient than the initial authentication.

In the rest of this section, we attempt to recognize the remaining requirements for ubiquitous access control. Each occurrence of access control consists of two principal phases: access rights of a user requesting an access is authenticated; and then the access rules accompanying the authenticated access rights are executed. In the following clauses, we identify the requirements for each phase.

3. 2 Requirements in authenticating access rights

Requirements for general authentication have been clearly recognized, and they also apply to authentication of access rights, since there is no essential difference between authentication of identities and that of access rights [11]. Therefore, we can identify the following 4 requirements.

Completeness A user who received access rights through authorized procedures successes in proving her access rights.

Soundness Without access rights issued through authorized procedures, nobody is able to prove his access rights.

Non-transferability A user cannot transfer her access rights to others.

Revocability A granting entity of access rights is capable of revoking access rights ever issued. Once access rights are revoked, the holding user of the access rights can no longer succeed in proving the revoked access rights.

3. 3 Requirements in executing access rules

The most significant difference of access control from authentication of identities is that access rights are necessarily accompanied by access rules.

Enforcement of access rules At the same time of authentication of access rights, the integrity of the accompanying access rules is to be verified. And then the verifying entity executes the access rules.

Authorized change of access rules Access rules are dynamic. Hence, access rules may contain rules to change themselves (*e.g.* one-time or consumable access rights).

In ubiquitous computing, we cannot always assume that the entity that granted access rights as well as access rules can control change of the access rules after the issuance, since authentication of the access rights and the consequent change of the access rules are executed off-line from the granting entity. Therefore, it is necessary that only authorized verifiers are capable of changing the access rules in the way specified in the access rules.

3. 4 Conclusion of Section 3.

As stated above, we have identified the following 10 requirements for ubiquitous access control. (1) *completeness*, (2) *soundness*, (3) *non-transferability*, (4) *revocability*, (5) *enforcement of access rules*, (6) *authorized change of access rules*, (7) *unlinkability*, (8) *consensual open*, (9) *verifier authentication*, and (10) *efficiency*. In Section 4., we will present a sketch of a scheme of access control protocols that supports the identified requirements.

4. The proposed scheme

4. 1 Players

The access control protocol presented in this section is designed assuming 4 players: *User Agent* (*UA*), *Service Provider* (*SP*) *Service Provider Agent* (*SP-A*), and *Service Appliance* (*SA*). It is also assumed that these players are totally independent of each other and any two of them independently determine whether to trust each other only based on their mutual agreement.

SP-A	*SP-A* is a monolithic module existing in a computing device that a user carries. *SP-A* helps the user to gain and prove her access rights to services and at the same time prevents her from abusing the access rights.
UA	Any instance of *UA* is a module also existing in the user's computing device, and is under the full control of the user. *UA* plays the role of supervising *SP-A* so that *SP-A* does not leak any data that breaks unlinkability.
SP	*SP* is an owner of services and is the only entity eligible to grant users access rights to the services. *SP* communicates with *SP-A* via *UA*to grant access rights.
SA	*SA* is a software program, device, apparatus and the like ubiquitously existing in the environment, and renders services on behalf of *SP*. *SA* communicates with *SP-A* via *UA* to verify access rights granted to the user who carries *SP-A*.

In particular, the relation between *UA* and *SP-A* is designed based on the *wallet-with-observer model* [12]. Therefore, *SP-A* (observer) is a tamper resistant module, configured so that all the communication with the outside goes through *UA* (wallet). The deployment of this model is appropriate for the following reasons.

- The requirement of non-transferability is desirable to be realized using prevention technologies instead of after-the-fact technologies. This is partly because the effectiveness of non-transferability due to after-the-fact technologies is doubtful when it is applied to very important services and resources. From a performance point of view, after-the-fact technologies are in general less efficient. For example, *all-or-nothing non-transferability* presented in [9] is known to require impractical amount of computation to execute it.
- The requirement of verifier authentication requires existence of an agent playing on behalf of *SP* at the user's point because of the off-line scenario (3. 3). Otherwise, *SA* and the user may collude with each other to cheat *SP*.

4. 2 Phases

The execution of the protocols of this paper is comprised of the following 3 phases.

a) Identifying services

SP generates a public key pair and assigns it to a service that it intends to provide to users. Access rights to the service is verified using the public key of the key pair (*service public key*), while *SP* uses the private key (*service private key*) to grant access rights to users.

b) Granting access rights

On request from a user, *SP* mathematically transforms the requested public private key to *Access ID* (4. 3. 2), and issues it to the requesting user. The transformation is one-time and one-way with a trap door.

c) Verifying access rights

For unlinkable proof of access rights, a user (*i.e. UA*) presents

Anonymized Access ID (4. 3. 3) to *SA*. For consensual open, Access ID is presented instead of Anonymized Access ID. *SA* verifies the user's access rights using the service public key and (Anonymized) Access ID.

4. 3 **Protocol specification**

4. 3. 1 Notations

In the rest of this section, the following notation will be used.

$\mathcal{G}_T$	A DLP-hard additive groups used for granting access rights. DLP-hardness is the property that solving $A \overset{\mathcal{G}_T}{=} xB$ in x is intractable.
$\mathcal{G}_S$	A DLP-hard additive groups used for verifying access rights.
$x \overset{\mathcal{G}_T}{=} y$	x and y are identical with each other in $\mathcal{G}_T$.
G_T (G_S)	The base element in $\mathcal{G}_T$ ($\mathcal{G}_S$, resp.) with orders n_T (n_S, resp.)
(T, τ)	A public key pair of *SP-A* such that $T \overset{\mathcal{G}_T}{=} \tau G_T$.
(S, σ)	A service public key pair such that $S \overset{\mathcal{G}_S}{=} \sigma G_S$.
(A, α)	A public key pair of *SA* such that $A \overset{\mathcal{G}_S}{=} \alpha G_S$
$\pi(x)$	A bijection that transforms x of $\mathcal{G}_T$ or $\mathcal{G}_S$ to a fixed-length bit string.
$\omega(x)$	A pseudo random function, which takes a variably long bit string as input and outputs a bit string of a fixed length. Pseudo-randomness implies being one-way and collision-free.
$\mu(\textit{key}, x)$	A secure MAC generation (verification) function.

4. 3. 2 Access ID

Access ID is data that *SP* issues to a user as representation of the access right that it grants to the user. Access ID, denoted by *aid*, is generated through cooperation between *SP* and *SP-A*. In fact, *aid* is the private key σ assigned to the *Service* masked with a random secret k shared between *SP* and *SP-A*.

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

4. 3. 3 Anonymized Access ID

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

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

4. 4 **Primitive protocols**

The protocols of this paper are comprised of the following primitive protocols.

Primitive protocol	Unlinkable	Non-anonymous
Rights granting	✓	
Rights verification	✓	✓
Rights consumption	✓	✓
Rights update	✓	✓
Rights revocation	✓	✓
Key transfer	✓	✓
Continual rights verification	✓	✓

4. 4. 1 Unlinkable rights granting protocol

Figure 1 depicts the unlinkable rights granting protocol. In the protocol, *SP* and *SP-A* shares a secret k according to the *unilateral* version of the MQV (Menezes-Qu-Vanstone) key sharing protocol [13]. The public key T used in the key sharing represents a group of *SP-A* and every member of the group shares the same (T, τ). What *SP* presumes from T is only whether the implementation of *SP-A* of the group is accredited to clear certain safety criteria. Since T belongs to a group of *SP-A*, *SP* cannot distinguish between instances of *SP-A* from T. In addition, since *UA* randomly generates ϵ_U, E_U uniformly distributes over $\mathcal{G}_T$. This is the reason why the rights granting protocol is unlinkable.

On the other hand, only the instance of *SP-A* that knows ϵ_T and ϵ_U can actually calculate k. Therefore, the attack where malicious *SP-A*'s are programmed to generate common ϵ_T to share k between them is not effective.

The data $*$ denotes a hash value of *access rules* accompanying *aid*, while the data $\#$ does an identifier of *aid* to be specified in *Rights Revocation List*. Both of them are bound to *aid* in a manner such that, if they are changed after the issue by *SP*, *SA* always fails in verification of *aid* or *anm* derived from it.

4. 4. 2 Unlinkable rights verifying protocol

Figure 2 depicts the unlinkable rights verifying protocol. *SA* verifies the response r received from *UA* by Eq. (1).

$$rG_S \overset{\mathcal{G}_S}{=} \omega(\pi(W) \mid c \mid *)(S - \textit{anm}G_S) + W \qquad (1)$$

The protocols presented in Figure 2 – 5 are designed based on the signature scheme derived from the Schnorr identification scheme [14], which is known to be *perfect* ZKIP (zero-knowledge interactive proof). Soundness of the protocols is to non-malleability of the underlying signature scheme — even if an attacker is so strong that he can exploit the signer as a signing oracle, he cannot generate signatures to new messages.

Unlinkability of this protocol is proved as follows.

If *UA* successfully verifies r', then unlinkability is derived from the fact that the protocol is ZKIP: new knowledge that *SA* can acquire from the transcript of the communication with *UA* is only *anm*, which uniformly distributes over $[0, n_S)$ due to random $w'' \in_R [0, n_S)$.

If *UA* fails in verification of r', *SA* only acquires two independent random number *anm* and W.

Also, since (W, r) is a digital signature to $c \mid *$ verifiable by the public key $S - \textit{anm}G_S$, *SP* can verify the integrity of the data $*$ in the sense that the data $*$ that *SP-A* receives is identical with the data $*$ that *SP* used to calculate *aid* and the data $*$ that *SA* sent to *SP-A* via *UA*.

4. 4. 3 Non-anonymous rights verifying protocol

Figure 3 depicts the non-anonymous rights verifying protocol.

The differences from the unlinkable counterpart are as follows.

- The messages exchanged between *SA* and *SP-A* pass through *UA* as they were when generated.
- *SP-A* calculates a as $\omega(\pi(W) \mid c \mid \mathit{key})$.

Since *key* is a secret between *SA* and *SP-A*, *SA* can verify r based on *aid*, but *UA* cannot present *anm* instead of *aid*: *UA* must change r in order to make *SA* succeed in verification of r based on *anm*, but cannot since *UA* doesn't know a.

4.4.4 Unlinkable and non-anonymous rights consumption protocols

The unlinkable and non-anonymous rights consumption protocols are respectively the same as the linkable and non-anonymous rights verification protocols except:

- after receiving r from *UA*, *SA* sends either $e_3 = \mu(\mathit{key}, \texttt{success})$ or $e_3 = \mu(\mathit{key}, \texttt{fail})$ depending on whether *SA* succeeded in verifying r;
- *SP-A* erases k from its internal database, if, and only if, it successfully verifies $e_3 = \mu(\mathit{key}, \texttt{success})$.

Erasing k means that *SP-A* can no longer generate valid responses r, and therefore *UA* cannot generate a proof of possession of *aid* any more.

4.4.5 Unlinkable and non-anonymous rights update protocols

Figure 4 depicts the unlinkable version of the rights update protocol. The differences from the anonymous rights consumption protocol is as follows:

- *SA* calculates and sends $\bar{*}$, which is a hash of the updated access rules;
- *SP-A* defines $a = \omega(\pi(W) \mid c \mid * \mid \bar{*})$ and calculates r' based on a;
- If e_3 asserts that *SA* succeeded in verifying r, *SP-A* generates new random $\bar{k}$ and calculates $\Delta = \mu(k, *) - \mu(\bar{k}, \bar{*})$. Further, *SP-A* erases k to revoke *aid*.

The new *aid* that *UA* calculates by $\mathit{aid} + \Delta \bmod n_S$ is accompanied by $\bar{*}$, which is a hash of the updated access rules. Since (W, r) is a digital signature to $c \mid * \mid \bar{*}$, if *UA* altered $\bar{*}$, *SA* would fail in verifying r and then *SP-A* would not calculate Δ.

4.4.6 Unlinkable and non-anonymous key transfer protocols

Key transfer is the feature for *SP* to transfer a secret key to *SA* so that *SA* receives it, if, and only if, *SA* succeeds in verifying the possession of k by *SP-A* and *SP-A* succeeds in authenticating *SA*. The function of key transfer makes supporting of the requirement of verifier authentication more assured, for example, in a case where the service contents are encrypted by the key to be transferred.

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

Figure 5 depicts the linkable key transfer protocol.

In performing the protocol, *SA* shall not send the raw L, since *UA* could also reveal K. Instead, *SA* selects a random $\lambda \in [0, n_S)$, and then sends $C \overset{\mathcal{G}_S}{=} \lambda L$ to *UA*. On receipt of R from *UA*, *SA* recaptures K by $(\lambda^{-1} \bmod n_S)K'$, since $K' \overset{\mathcal{G}_S}{=} \mathit{anm} \cdot C + R \overset{\mathcal{G}_S}{=} \sigma C \overset{\mathcal{G}_S}{=} \lambda\kappa S$.

To supports unlinkability, *UA* shall verify that R' is identical with $\mu(k, *)C$ without knowing $\mu(k, *)$. Otherwise, *SP-A* could send arbitrary data, which may reveal the identity of the user. For this purpose, we deploys the well known technique for investigating a decisional Diffie-Hellman tuple, and hence *UA* verifies Eq. (2).

$$V \overset{\mathcal{G}_S}{=} x(S - \mathit{aid} \cdot G_S) + yR' \qquad (2)$$

In fact, if Eq. (2) holds, the equation $\log_{G_S} V \equiv \mu(k, *)x + \left(\log_{G_S} R'\right) y \bmod n_S$ holds. On the other hand, $U \overset{\mathcal{G}_S}{=} xG_S + yC$ implies the equation $\log_{G_S} U = x + \lambda\kappa y$. Therefore, unless $\log_{G_S} R' \equiv \lambda\kappa\mu(k, *) \bmod n_S$, only a single (x, y) out of n_S^2 candidates makes both of the equations hold. Apparently, the probability that this case happens is vanishingly small.

The rest of the proof of unlinkability of the unlinkable key transfer protocol is the same as that of the unlinkable rights verification protocol.

4.4.7 Unlinkable and non-anonymous rights revocation protocols

To support revocability, it suffices that *SP* sends *Rights Revocation List*, which specifies a list of the $\#$ identifiers of *aid*'s to be revoked, and *SP-A* erases k corresponding to the entries of the list. To prevent the list from being tampered, *SP-A* calculates r' so that (W, r') is a signature of data containing the list.

4.5 Conclusion of Section 4.

The protocols presented above supports all the requirements presented in Section 3.. In fact, (3) non-transferability is due to the design that *SP-A* securely reserves secret k shared with *SP*. The requirements (1), (2) and (4) to (10) are supported due to the functionalities of the primitive protocols.

Protocols	(1)	(2)	(4)	(5)	(6)	(7)	(8)	(9)
Anonymous version						√		
Non-anonymous version							√	
Rights verification	√	√		√				√
Rights consumption					√			
Rights update					√			
Key transfer								√
Rights revocation			√					

(1) Completeness (2) Soundness (4) Revocability
(5) Enforcement of access rights
(6) Authorized change of access rules (7) Unlinkability
(8) Consensual open (9) Verifier authentication

With respect to efficiency (10), our protocols are more efficient than the schemes by [8] and [15]. In fact, our anonymous rights verification protocol require 6-time execution of the scalar multiplication on an elliptic curve, while the scheme [8] or [15] does 22-time execution of the modular exponentiation.

In addition, continual authentication can be efficiently executed using *key* shared between *SA* and *SP-A* at the initial execution of the protocol.

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*, pp. 709–712, 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*, pp. 129–136, 2003.

[4] D.Chaum and E. van Heist, "Group signatures," in *EUROCRYPT 1991, LNCS 547*, pp. 257–265, 1991.

[5] J. L. Camenisch and M. A. Stadler, "Efficient group signature schemes for large groups," in *CRYPTO 1997, LNCS 1294*, pp. 410–424, 1997.

[6] L. Chen and T. P. Pedersen, "New group signature schemes," in *EUROCRYPT 1994, LNCS 550*, pp. 171–181, 1995.

[7] D. X. Song, "Practical forward secure group signature schemes," in *ACM Conference on Computer and Communications Security*, pp. 225–234, 2001.

[8] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik, "A practical and provably secure coalition-resistant group signature scheme," in *CRYPTO 2000, LNCS 1880*, pp. 255–270, 2000.

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

[10] 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), pp. 21–30, ACM Press, 2002.

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

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

[13] 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*, pp. 22–32, 1995.

[14] C. Schnorr, "Efficient identification and signatures for smart cards," in *CRYPTO 1989, LNCS 435*, pp. 239–252, 1990.

[15] J. Camenisch and A. Lysyanskaya, "An efficient system for non-transferable anonymous credentials with optional anonymity revocation," in *EUROCRYPT 2001, LNCS 2045*, pp. 93–117, 2001.

SP		*UA*		*SP-A*
			$\xleftarrow{E_T}$	$\epsilon_T \in_R [0, n_T)$ $E_T \overset{\mathcal{G}_T}{=} \epsilon_T G_T.$
		$\epsilon_U \in_R [0, n_T)$ $E_U \overset{\mathcal{G}_T}{=} \epsilon_U G_T + E_T$		
$\epsilon_P \in_R [0, n_T)$ $E_P \overset{\mathcal{G}_T}{=} \epsilon_P G_T$ $k' = \pi(\epsilon_P(E_U + \bar{E}_U T))$ $k = \mu(k', \#)$ $\mathit{aid} = \sigma - \mu(k, *) \bmod n$	$\xleftarrow{E_U}$			
	$\xrightarrow{\mathit{aid}, *}$	Store *aid* and $*$.		
	$\xrightarrow{E_P, \#}$	$\longrightarrow$	$\xrightarrow{E_P, \#, \epsilon_U}$	$E_U \overset{\mathcal{G}_S}{=} (\epsilon_U + \epsilon_T) G_S$ $k' = \pi((\epsilon_U + \epsilon_T + \bar{E}_U \tau) E_P)$ $k = \mu(k', \#)$ Store k and #.

Fig. 1 Rights granting protocol

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

Fig. 2 Anonymous rights verifying protocol

SA		*UA*		*SP-A*
	$\xleftarrow{aid}$			
$c \in_R [0, n_S), e_0 = \omega(c)$	$\xrightarrow{e_0}$	$\longrightarrow$	$\xrightarrow{e_0}$	
key $= \omega(\pi(\alpha W) \mid c)$	$\xleftarrow{W}$	$\longleftarrow$	$\xleftarrow{W}$	$w \in_R [0, n_S), W \overset{\mathcal{G}_S}{=} wG_S$
$e_1 = \omega($*key*$)$	$\xrightarrow{c, e_1, x}$	$\longrightarrow$	$\xrightarrow{c, e_1, x}$	*key* $= \omega(\pi(wA) \mid c)$ Unless e_0 and e_1 are valid, abort the session. $a = \omega(\pi(W) \mid c \mid$*key*$) \bmod n_S$ $r = a\mu(k, *) + w \bmod n_S$
Unless r and e_2 are valid, abort the session.	$\xleftarrow{r}$	$\longleftarrow$	$\xleftarrow{r}$	

Fig. 3　Non-anonymous rights verifying protocol

SA		*UA*		*SP-A*
		The same as the rights authentication protocol.		
$e_1 = \omega($*key*$)$	$\xrightarrow{c, e_1, *, \bar{*}}$			
			$\xrightarrow{c, e_1, *, \bar{*}, w''}$	*key* $= \omega(\pi((w' + w'')A'') \mid c)$ Unless e_0 and e_1 are valid, abort the session. $W \overset{\mathcal{G}_S}{=} W' + w''G_S$ $a = \omega(\pi(W) \mid c \mid * \mid \bar{*}) \bmod n_S$
			$\xleftarrow{r'}$	$r' = a\mu(k, *) + w' \bmod n_S$
		Unless r' is valid, abort the session. $a = \omega(\pi(W) \mid c \mid * \mid \bar{*}) \bmod n_S$ $r = r' + a\rho + w'' \bmod n_S$		
Unless r is valid, abort the session.	$\xleftarrow{r}$			
$e_3 = \mu($*key*$, 1)$	$\xrightarrow{e_3}$	$\longrightarrow$	$\xrightarrow{e_3}$	Unless e_3 is valid, reject the session. Erase k from Key Registry. Generate new random $\bar{k}$ and store it in Key Registry.
			$\xleftarrow{\Delta}$	$\Delta = \mu(k, *) - \mu(\bar{k}, \bar{*}) \bmod n_S$
		Replace *aid* with *aid* $+ \Delta$		

Fig. 4　Anonymous rights update 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 $= \omega(\pi(\alpha W) \mid c)$	$\xleftarrow{W}$			
$e_1 = \omega($*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* $= \omega(\pi((w' + w'')A) \mid c)$ Unless e_0 and e_1 are valid, abort the session. $\underline{R' \overset{\mathcal{G}_S}{=} \mu(k, *)C}$ $\underline{V \overset{\mathcal{G}_S}{=} \mu(k, *)U}$ $a = \omega(\pi(W) \mid c \mid *) \bmod n_S$
			$\xleftarrow{r', R', V}$	$r' = a\mu(k, *) + w' + w'' \bmod n_S$
		Unless r' and R' are valid, abort the session. $a = \omega(\pi(W) \mid c \mid *) \bmod n_S$ $r = r' + a\rho \bmod n_S$ $\underline{R \overset{\mathcal{G}_S}{=} R' + \rho C}$		
Unless r is valid, abort the session. $\underline{K' \overset{\mathcal{G}_S}{=} \textit{anm} \cdot C + R}$	$\xleftarrow{r, R}$			

Fig. 5　Key transfer protocol