情報保護學會論文誌
第 21 卷 第 4 號, 2011. 8

# 짧은 그룹 서명 기법 기반의 익명 인가에 대한 연구*

신 수 연†, 권 태 경‡
세종대학교 컴퓨터공학과

# A Study on Anonymous Authorization based on Short Group Signatures*

Sooyeon Shin† and Taekyoung Kwon‡
Department of Computer Science, Sejong University

요 약

프라이버시 보호를 위한 그룹 서명 기법에 기반한 기존의 익명 인증 기법은 실제 응용 환경에서 필요로 하는 익명 인가는 제공하지 못한다. 이를 해결하기 위해서 본 논문에서는 짧은 그룹 서명 기법에 기반하여 익명 인증과 동시에 익명 인가를 통해 사용자에게 권한별 서비스 제공이 가능한 익명 인증 및 인가 기법을 제안한다. 익명 인증 및 인가 기법은 그룹 매니저의 권한 분할과 권한 매니저를 이용하여, 실명, 익명, 권한이 모두 분리 관리 되도록 하며, 다양한 접근제어 모델의 적용이 가능하다.

ABSTRACT

The existing anonymous authentication schemes based on group signatures for protecting privacy do not provide anonymous authorization which is required in the practical environments. In this paper, we propose an anonymous authentication and authorization scheme that enables a service provider both to authenticate anonymously its users and to provide different service according to their authorization. In the proposed scheme, a user's real identity, anonymity and authorization are managed distinctly through the separation of group manager's capabilities and an authorization authority. It is also possible for the proposed scheme to apply various access control models.

**Keywords**: privacy preservation, anonymous authentication, anonymous authorization, fine-grained authorization

## I. 서 론

인터넷의 보급이 일반화 되어감에 따라 사용자의 개인 정보가 유출되는 프라이버시 침해 문제가 대두되고 있으며, 유비쿼터스 환경이 도래함에 따라 개인 정보 유출의 문제는 더 심각해지고 있다. 이러한 문제를 해결하기 위해 최근 민간단체와 국제기구를 중심으로 한 프라이버시 보호 기술(PETs: Privacy Enhancing Technologies) 개발이 활발히 이루어지고 있으며, W3C와 ISO/ICE에서 프라이버시 보호 표준 기술이 연구되고 있다. 또한 인터넷의 여러 곳에 흩어진 개인 정보를 통합적으로 관리 및 보호할 수 있도록 IDM (Identity Management)에서도 익명화 기법과 프라이버시 보호 접근 제어 (privacy-preserving aware access control) 등과 같은 프라이버시 보호를 위한 기술을 연구 중에 있다.

프라이버시 보호를 위한 대표적인 기술은 익명성 제공 기술로써 사용자를 익명으로 인증하고 서비스를 제공할 수 있도록 한다. 익명 인증은 은닉 서명, 링 서명, 그룹 서명 등 전자 서명 기법을 기반으로 제공될 수 있다. 하지만 익명 사용자의 욕설이나 비방 글 게

접수일(2010년 11월 8일), 게재확정일(2011년 3월 15일)
* 이 논문은 2011년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. 2011-0002467).
† 주저자, shinsy80@sju.ac.kr
‡ 교신저자, tkwon@sejong.ac.kr

시 등의 불법적인 행위는 사회적으로 문제가 되고 있으므로, 무분별한 인터넷 서비스 오남용을 방지하기 위해서는 실명 정보를 추적할 수 있는 기술이 필요하다. 그룹 서명 기법은 익명 인증을 지원하면서 서명자의 신원을 추적할 수 있는 기능을 제공한다.

Chaum과 van Heyst가 1991년에 소개한 그룹 서명 기법은 사용자가 신원을 노출시키지 않으면서 그룹에 속해 있음을 증명할 수 있는 기술이다[8]. 그룹의 모든 멤버가 메시지에 서명하는 것이 가능하지만, 그룹 관리자(GM: Group Manager) 외에 누구도 서명으로부터 서명자의 신원 정보를 유도하는 것은 불가능하므로 익명성을 제공한다. Chaum과 van Heyst가 그룹 서명을 소개한 이후, 다양한 그룹 서명 기법과 그룹 서명 기법을 기반으로 한 익명 인증 기법이 제안되어 왔다[9], [5], [6], [1], [13], [4], [7]. 대부분의 익명 인증 기법은 실제 응용에 필요한 접근 통제를 위한 인가는 제공하지 못한다.

본 논문에서는 그룹 서명 기반의 익명 인증과 동시에 익명 인가가 가능한 기법을 제안한다. 제안하는 기법은 그룹 서명의 그룹 관리자의 권한을 분리하여 조건부 추적이 가능하도록 하며, 권한 관리자를 두어 사용자의 실명, 익명, 권한[1]이 모두 독립적으로 관리되도록 한다. 제안하는 익명 인가 기법은 권한을 구분할 수 있는 랜덤 값을 사용하므로 다양한 접근 제어 모델을 적용하는 것이 가능하며, 익명 인가를 위한 값을 이용하여 익명성이 취소된 후, 사용자의 이전 트랜잭션 추적가능 서명 기법으로도 활용 가능하다.

본 논문은 다음과 같이 전개된다. 2장에서는 관련 연구에 대해 간략히 정리하며, 3장에서는 제안 기법의 가정 및 정의를 설명한다. 4장에서는 제안하는 짧은 그룹 서명 기반의 익명 인증 및 인가 기법에 대해 소개하고, 5장에서는 제안하는 기법의 안전성 및 성능에 대해 설명한다. 마지막으로, 6장에서는 결론과 향후 계획에 대해서 논의한다.

## II. 관련연구

Chaum과 van Heyst가 1991년에 소개한 그룹 서명 기법은 그룹의 멤버로 등록한 사용자가 신원을 노출시키지 않으면서 그룹 서명 생성이 가능하며, 이 서명을 통해 그룹에 속해 있음을 증명할 수 있는 기술이다[8]. Chen과 Peterson[9], Camenisch[5] 등이 확장 시켰으며, 최근까지 활발히 연구되고 있다. 위에서 언급한 초기 그룹 서명 기법의 경우에는 그룹의 크기가 커지면 서명의 크기가 커져 크기가 큰 그룹에는 적합하지 않았다. 이러한 문제를 해결하기위해 Camenisch와 Stadler[6], Ateniese 등 [1]., Nguyen과 Safavi-Naini[13]는 그룹 크기와 상관없이 고정된 크기의 그룹 공개키와 서명을 가지는 그룹 서명 기법을 제안하였다. 이전의 대부분의 기법들은 Strong RSA 가정에 기반한 기법들이였으며, 2004년 Boneh 등은 SDH (Strong Diffie-Hellman) 가정과 겹선형 쌍함수(bilinear pairing)에 기반하여 짧은 길이의 서명을 생성하는 짧은 그룹 서명 기법을 제안하였다[4]. Camenisch와 Lysyanskaya는 SDH 가정과 유사한 이산대수 형태의 LRSW (Lysyanskaya, Rivest, Sahai, Wolf) 가정[12]에 기반한 그룹 서명 기법을 제안하였다[7].

2004년에 Kiayias 등은 그룹 서명 기법에서 그룹 매니저 혹은 공개자가 서명자의 신원을 노출시키는 Open 기능만으로는 프라이버시 보호 측면에서 충분하지 못하다는 점을 지적하였다. 그룹 서명의 이러한 점을 보완하기위해 특정 멤버가 서명한 서명들을 탐지하여 해당 멤버의 트랜잭션(transaction)을 모두 추적 가능한 추적가능 서명 기법을 제안하였다[11]. 2006년 Choi 등은 Boneh 등이 제안한 짧은 그룹 서명 기법을 이용하여 Kiayian 등의 추적가능 서명 기법의 서명 길이를 1/3정도로 줄였다[10].

그룹 서명 기법과 추적가능 서명 기법은 모두 익명성을 추적 및 공개하는 것이 가능한 기법으로 익명 인증을 위해 사용하는 것이 가능하다. 실제 응용에서 인가는 기본적으로 식별 및 인증과 함께 중요시되는 보안 기술 중 하나이다. 하지만, 그룹 서명 기법과 추적가능 서명 기법은 사용자의 익명성을 유지함과 동시에 인가를 제공하지 못한다.

이러한 문제를 해결하기 위해, 익명 인증과 동시에 익명 인가를 제공하기 위한 연구가 이루어졌다. 2007년 Benjumea 등은 X.509 프레임워크에 새로운 익명 서명 기법을 통합하여 X.509 인증서를 이용하여 익명 인증이 가능하도록 하였다[2]. X. 509 공개키 인증서는 한 개체의 개인키와 연결된 공개키를 포함하여 해당 개체의 인증이 가능하도록 한다. 하지만, Benjumea 등은 X.509 공개키 인증서가 가지는 공개키를 하나의 개체가 아닌 다수의 개체의 개인키와 연결이 가능하도록 하였다. 즉, 하나의 X.509 공개키

1) 본 논문에서는 사용자의 속성 정보 또한 접근 제어를 위한 권한으로 보며, 이후 속성을 권한에 포함하여 전개한다.

프로토콜 Z

단계 1. 증명자는 $T_1, T_2, T_3, T_4, T_5$ 계산

먼저 지수 값 $k', \alpha, \beta$ 을 랜덤하게 $\mathbb{Z}_p$에서 선택하고, $\delta_1 \leftarrow x\alpha, \delta_2 \leftarrow x\beta, \delta_3 \leftarrow \tau\alpha, \delta_4 \leftarrow \tau\beta \in \mathbb{Z}_p$를 계산한다. 해당 값들을 이용하여 $A$의 선형 암호인 $T_1, T_2, T_3$와 $T_4, T_5$를 계산한다.

$$T_1 \leftarrow u^{\alpha}, T_2 \leftarrow v^{\beta}, T_3 \leftarrow Ah^{\alpha+\beta}, T_4 \leftarrow g_1^{k'}, T_5 \leftarrow e(T_4, g_2)^{\tau}$$

단계 2. 증명자와 검증자는 다음의 관계를 만족하는 값 $(\alpha, \beta, x, \tau, \delta_1, \delta_2, \delta_3, \delta_4)$의 지식 증명을 착수

$$u^{\alpha} = T_1,\ v^{\beta} = T_2,\ T_1^{x} u^{-\delta_1} = 1,\ T_2^{x} v^{-\delta_2} = 1,\ T_1^{\tau} u^{-\delta_3} = 1,\ T_2^{\tau} v^{-\delta_4} = 1,\ e(T_4, g_2)^{\tau} = T_5,$$
$$e(T_3, g_2)^{x} \cdot e(T_3, w)^{\tau} \cdot e(h, g_2)^{-\delta_1 - \delta_2} \cdot e(h, w)^{-\delta_3 - \delta_4} = e(g_1, g_2)$$

단계 2-1. 증명자는 은닉 값(blinding values) $r_{\alpha}, r_{\beta}, r_x, r_{\tau}, r_{\delta_1}, r_{\delta_2}, r_{\delta_3}, r_{\delta_4}$을 랜덤하게 $\mathbb{Z}_p$에서 선택하고 $R_1, \cdots, R_8$을 다음과 같이 계산

$$R_1 \leftarrow u^{r_{\alpha}},\ R_2 \leftarrow v^{r_{\beta}}, R_4 \leftarrow T_1^{r_x} u^{-r_{\delta_1}}, R_5 \leftarrow T_2^{r_x} v^{-r_{\delta_2}}, R_6 \leftarrow T_1^{r_{\tau}} u^{-r_{\delta_3}}, R_7 \leftarrow T_2^{r_{\tau}} v^{-r_{\delta_4}}, R_8 \leftarrow e(T_4, g_2)^{r_{\tau}},$$
$$R_3 \leftarrow e(T_3, g_2)^{r_x} \cdot e(T_3, w)^{r_{\tau}} \cdot e(h, g_2)^{-r_{\delta_1} - r_{\delta_2}} \cdot e(h, w)^{-r_{\delta_3} - r_{\delta_4}}$$

단계 2-2. 증명자가 검증자에게 $(T_1, \cdots, T_5, R_1, \cdots, R_8)$을 전송

단계 2-3. 검증자는 $\mathbb{Z}_p$에서 랜덤하게 선택된 도전 값(challenge) $c$를 전송

단계 2-4. 증명자는 검증자에게 받은 도전 값을 이용하여 $s_{\alpha}, s_{\beta}, s_x, s_{\tau}, s_{\delta_1}, s_{\delta_2}, s_{\delta_3}, s_{\delta_4}$을 계산하여, 검증자에게 계산 결과를 전송

$$s_{\alpha} = r_{\alpha} + c\alpha, s_{\beta} = r_{\beta} + c\beta, s_x = r_x + cx, s_{\tau} = r_{\tau} + c\tau, s_{\delta_1} = r_{\delta_1} + c\delta_1, s_{\delta_2} = r_{\delta_2} + c\delta_2, s_{\delta_3} = r_{\delta_3} + c\delta_3, s_{\delta_4} = r_{\delta_4} + c\delta_4$$

단계 2-5. 검증자는 다음의 식 (1) - (8)을 검사하여, 모든 식이 만족한다면 검증자는 수락하고, 그렇지 않다면 거절

$$u^{s_{\alpha}} = T_1^{c} \cdot R_1 \quad (1) \qquad v^{s_{\beta}} = T_2^{c} \cdot R_2 \quad (2)$$
$$e(T_3, g_2)^{s_x} \cdot e(T_3, w)^{s_{\tau}} \cdot e(h, g_2)^{-s_{\delta_1} - s_{\delta_2}} \cdot e(h, w)^{-s_{\delta_3} - s_{\delta_4}} = e(g_1, g_2)^{c} \cdot R_3 \quad (3)$$
$$T_1^{s_x} u^{-s_{\delta_1}} = R_4 \quad (4)\ T_2^{s_x} v^{-s_{\delta_2}} = R_5 \quad (5)\ T_1^{s_{\tau}} u^{-s_{\delta_3}} = R_6 \quad (6)\ T_2^{s_{\tau}} v^{-s_{\delta_4}} = R_7 \quad (7)\ e(T_4, g_2)^{s_{\tau}} = R_8 \cdot T_5^{c} \quad (8)$$

인증서를 통해 그룹의 서로 다른 개인키를 가진 모든 멤버가 인증을 받을 수 있도록 하였다. 해당 인증은 개체가 해당 그룹의 정당한 멤버 인지를 검증하여 이루어지며, 개체의 신원은 노출되지 않으므로 익명성을 제공한다. Benjumea 등은 X.509 공개키 인증서에 하나 이상의 속성 인증서를 연결하여 권한 정보를 포함할 수 있도록 하여 익명 인가를 제공하였다. 속성 인증서 또한 하나의 X.509 공개키 인증서를 공유하는 모든 개체가 공유하도록 하여 비연결성을 제공하였다.

또 다른 익명 인가에 관련 연구는 2009년 Yao와 Tamassia에 의해 이루어졌다[14]. Yao와 Tamassia는 익명 서명자의 집성 서명(aggregate signature)을 사용하여 역할 기반 접근제어 기법을 사용하는 환경에서 역할을 기반으로 권한 체인을 생성하고 역할별로 서로 다른 권한을 가지도록 하여 특정 역할을 가지는 개체가 다른 역할을 가진 개체에게 신원 정보를 노출하지 않은 상태로 권한을 위임할 수 있도록 하였다. 각 역할의 멤버는 역할 인증서를 가지며 역할 인증서를 기반으로 익명 위임 인증서를 생성할 수 있다.

Benjumea 등과 Yao와 Tamassia의 기법은 익명 인가가 가능하도록 하지만 두 개의 기법 모두 특정 그룹 혹은 역할 별로 권한을 부여한다. 하지만, 서비스 제공자는 사용자의 속성과 권한에 따라 적합한 서비스를 제공하기를 원하는 경우가 많다. 그룹 서명 혹은 추적가능 서명 기법 기반 익명 인증과 위에서 언급한 익명 인가에 관한 연구들은 사용자가 특정 그룹의 정당한 멤버인지만을 판단하고 특정 그룹 혹은 역할이 가지는 권한만을 검증 가능하므로, 사용자 개개인이 가지는 특정 속성과 권한을 파악하기는 힘들다. 익명 인증과 함께 익명 인가를 위해서는 속성 및 권한이 해당 사용자의 것인지를 판단 할 수 있어야만 속성 및 권한의 위조를 막을 수 있으므로 익명 인증을 위해서 영지식 프로토콜을 사용하는 그룹 서명과 추적가능 서명 기법을 그대로 사용하는 것은 불가능하다. 따라서 익명성을 유지하면서 특정 익명 사용자의 권한을 비교하여 익명 인증과 동시에 사용자별로 가진 권한을 검증하여 익명 인가를 제공하는 기법이 필요하다.

## III. 가정 및 정의

### 3.1 겹선형 쌍함수(Bilinear pairing)

$G_1, G_2$는 위수가 소수 $p$인 곱셈 순환군으로 $g_1$과 $g_2$는 각각 $G_1$과 $G_2$의 생성원이다. $\psi$은 $\psi(g_2) = g_1$인 $G_2$

에서 $G_1$으로 계산 가능한 동형 사상(isomorphism)이다. $e$는 다음의 속성을 가지는 $e: G_1 \times G_2 \rightarrow G_T$가 다음의 조건을 만족할 때 겹선형 쌍함수라 한다.

Bilinearity: 모든 $u \in G_1, v \in G_2$과 $a, b \in \mathbb{Z}_p$에 대해, $e(u^a, v^b) = e(u,v)^{ab}$를 만족

Non-degeneracy: $e(g_1, g_2) \neq 1$이고 $e(g_1, g_2)$는 $G_T$의 생성원

### 3.2 복잡도 가정

$q$-Strong Diffie-Hellman 가정: $G_1, G_2$는 위수가 소수 $p$인 곱셈 순환군이고, $g_1$과 $g_2$는 각각 $G_1$과 $G_2$의 생성원일 때, $G_1 \times G_2 \rightarrow G_T$인 겹선형 사상에서 주어진 $(q+2)$-tuple $(g_1, g_2, g_2^{\gamma}, g_2^{(\gamma^2)}.., g_2^{(\gamma^q)})$로부터 $x \in \mathbb{Z}_p^*$인 쌍 $(g_1^{1/(\gamma+x)}, x)$을 계산하는 문제가 어려운 점을 기본 전제로 하는 가정

Decision Linear Diffie-Hellman 가정: $G_1$은 위수가 소수 $p$인 순환군이고, $u, v, h$는 $G_1$의 생성원일 때, 주어진 $u, v, h, u^a, v^b, h^c \in G_1$에서 $a+b$와 $c$를 구별하는 문제가 어려운 점을 기본 전제로 하는 가정

### 3.3 SDH 표현(representation)에 대한 영지식 프로토콜

익명 인증과 인가를 위해 짧은 그룹 서명 기법[4]의 영지식 프로토콜의 수정이 필요하며, 해당 프로토콜을 '프로토콜 Z'로 명명한다.

공개 값 $g_1, u, v, h \in G_1$과 $g_2, w \in G_2$을 가정한다. 여기서, $u, v, h$는 $G_1$의 랜덤한 값이고 $g_2$는 $G_2$의 랜덤한 생성원이다. $g_1$은 $\psi(g_2)$와 같으며, 비밀의 $\gamma \in \mathbb{Z}_p$에 대해 $w$는 $g_2^{\gamma}$와 같다. 다음의 프로토콜 Z는 $e(A, w^{\tau} g_2^{x}) = e(g_1, g_2)$를 만족하는 $(A, x, \tau)$의 소유를 증명한다. 여기서, $A \in G_1$, $x, \tau \in \mathbb{Z}_p$ 이고 $A^{x+\tau\gamma} = g_1$이다.

### 3.4 짧은 그룹 서명 기법의 안전성

Boneh 등은 짧은 그룹 서명 기법의 안전성을 보이기 위해, 다음의 속성을 만족시킴을 보였다[4].

- 정당성(correctness): 그룹 멤버가 생성한 서명은 올바르게 검증과 추적이 가능해야함
- 충분 익명성(full-anonymity): 서명은 서명자의 신원을 노출시키지 않아야함
- 충분 추적성(full-traceablility): 그룹 멤버가 생성한 서명으로부터 서명자 추적(open) 가능해야함

## IV. 제안 기법: 짧은 그룹 서명 기반 익명 인증 및 인가

그룹 서명 혹은 추적가능 서명 기법을 익명 인증을 위해 사용하는 것은 가능하지만, 익명 인가를 위해 사용하는 것은 불가능하다. 하지만 실제 응용 환경에서는 사용자에게 개인화된 서비스 제공 및 리소스의 접근 통제를 위해, 인가를 필요로 하므로 본 논문에서는 그룹 서명 기법에 기반한 익명 인증과 인가가 동시에 가능한 기법을 제안한다.

### 4.1 표기법

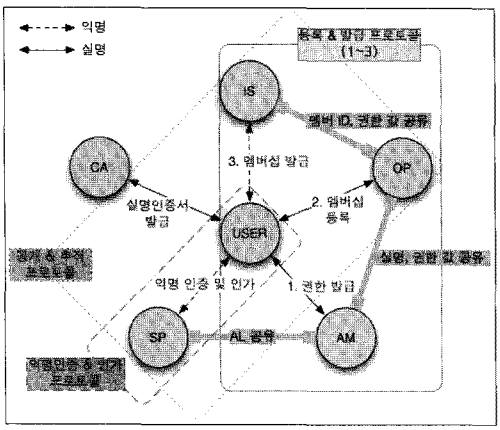
본 논문에서는 [표 1]의 표기법을 사용한다.

### 4.2 모델

앞서 언급한 것과 같이 그룹 서명 기법에서는 그룹 관리자(GM)라 불리는 특별한 권한을 가진 기관이 존재한다. GM은 사용자를 그룹의 멤버로 등록 받아 멤버쉽을 발급하고 추후에 특정 사용자의 익명성을 취소

[표 1] 표기법

| | |
|---|---|
| CA | PKI 인증서 발급기관 (Certificate Authority) |
| GM | 그룹 관리자(Group Manager)로 공개자(OP: Opener)와 발급자(IS: Issuer) 포함 |
| AM | 권한 관리자(Authorization Manager) |
| SP | 서비스 제공자(Service Provider) |
| $pub_i, pri_i$ | $i$의 PKI 공개키, 개인키 쌍 |
| $cert_i$ | $i$의 PKI 인증서(실명 인증서) |
| UN | 사용자의 실명(User Name) |
| $Sig_i$ | $i$의 서명 값 |
| $Msg_i$ | $i$의 메시지 |
| $E_{pub_i}$ | PKI기반 암호화 |
| $S_{pri_i}$ | PKI기반 서명 |
| $AL$ | 권한 리스트(Authorization List) |
| $AUTH$ | 접근 제어 모델에 따른 권한 집합 |
| $Iss_j^i$ | $i$에 대한 $j$의 발급(Issue) 트랜스크립트 |
| $J_j^i$ | $i$에 대한 $j$의 등록(Join) 트랜스크립트 |




〔그림 1〕 익명 인증 및 인가 모델

하기 위한 공개 프로토콜을 수행할 수 있는 능력을 가진다. 본 논문에서는 GM의 멤버쉽 발급과 공개 프로토콜 수행의 권한을 두 개의 기관으로 분산시킨다. 전자의 권한을 가지는 기관은 IS라 부르며, 후자의 권한을 가지는 기관을 OP라 부른다. 본 모델에서는 IS와 OP 외에 CA와 AM를 고려한다. CA는 사용자와 각 기관에게 실명 PKI 인증서 발급하는 기관이며, AM은 권한 발급을 위한 기관으로 사용자의 권한에 맞는 값을 할당하고 사용자를 익명으로 인가할 수 있도록 권한 값과 해당 권한 정보를 포함하는 권한 리스트(AL: Authorization List)를 SP에게 제공한다. OP는 사용자의 실명 인증서를 바탕으로 그룹에 등록시키고 AM이 발급한 권한 값 검증 역할을 한다. 또한, 추후에 서명으로부터 서명자에 대한 정보를 발견할 수 있는 공개 프로토콜의 일부를 수행한다. IS는 OP를 통해 그룹에 등록된 사용자에게 멤버쉽 비밀키를 발급하는 역할을 하며, 이 때 사용자의 실명은 알지 못한다. 따라서 사용자의 익명성 취소를 위해서는 반드시 OP와 IS, 두 기관이 협력해야만 한다. SP는 그룹 서명에서 검증자 역할을 하는 기관으로, 사용자를 익명으로 인증하고 권한을 검증하여 사용자에게 적합한 서비스를 제공하는 기관이다. 〔그림 1〕은 익명 인증 및 인가 모델을 보여준다.

## 4.3 익명 인증 및 인가 기법

익명 인증 시스템의 경우에는 사용자의 프라이버시 보호를 위해 그룹 서명 기법들을 기반으로 익명 인증이 이루어지지만, 접근 제어를 위한 권한 정보를 확인하여 각 사용자에게 적합한 서비스를 제공하는 것은 어렵다. 이를 해결하기 위해, 짧은 그룹 서명 기법에 기반 하여 익명 사용자의 권한 정보를 판단할 수 있는 익명 인증 및 권한 검증 기법을 제안하다. 본 기법은 키 생성, 권한 발급, 멤버 등록, 멤버쉽 발급, 익명 인가를 위한 메시지 생성, 익명 인증을 위한 서명 생성, 익명 인증 및 인가, 공개 프로토콜, 총 8개의 프로토콜과 내부 알고리즘으로 구성된다. 프로토콜 수행 이전, IS, OP, AM, 사용자는 CA로부터 공개키 기반 구조(PKI: public key infrastructure)에서 사용 가능한 공개키와 개인키 쌍을 생성하여 PKI 인증서 ($cert_i$)를 발급받았다고 가정한다.

### 4.3.1 키생성 프로토콜

그룹 관리자인 IS와 OP는 다음의 키 생성 알고리즘 $KeyGen$을 이용하여 함께 그룹 공개키 $gpk$와 각자의 비밀키 $sk_{IS}, sk_{OP}$를 생성하고, AM은 권한 공개키 $apk$ 를 생성한다.

| $KeyGen(1^k) = (gpk, sk_{IS}, sk_{OP}, apk)$ | |
|---|---|
| IS | $\gamma \in_R \mathbb{Z}_p^*,\ w = g_2^{\gamma}$ |
| OP | $h \in_R G_1 \setminus \{1_{G_1}\},\ \xi_1, \xi_2 \in_R \mathbb{Z}_p^*, u = h^{-\xi_1}, v = h^{-\xi_2}$ |
| $gpk = (g_1, g_2, h, u, v, w),\ sk_{IS} = (\gamma),\ sk_{OP} = (\xi_1, \xi_2)$ | |
| AM | $\pi \in_R \mathbb{Z}_p^*,\ apk = (\pi)$ |

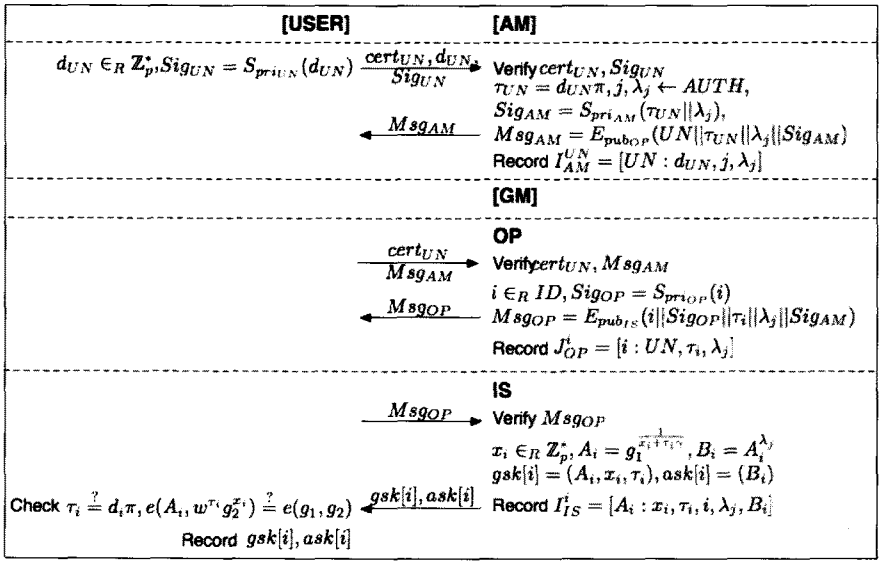
### 4.3.2 등록 및 발급 프로토콜

등록 및 발급 프로토콜은 권한 발급, 멤버 등록, 멤버쉽 발급 알고리즘을 포함하며, 〔그림 2〕는 해당 프로토콜을 보여준다.

4.3.2.1 권한 발급

사용자($UN$)는 그룹의 멤버로 등록하기 전, AM으로부터 인가를 위한 권한 정보를 발급받는다. 권한 발급을 위해 사용자는 AM에게 권한 가명(pseudonym)을 등록하고 권한 값을 받는다.

① 사용자는 $cert_{UN}$, 랜덤하게 선택한 가명 $d_{UN}$, $d_{UN}$의 서명 값을 가지고 AM에게 필요한 권한 값을 요청한다. $d_{UN}$의 서명 값은 사용자가 해당 가명에 대한 책임이 있음을 의미하며, 각 사용자의 $d_{UN}$는 추후 익명성 취소의 경우 취소 리스트에 포함되어 취소되지 않은 사용자의 멤버쉽 비밀 키 업데이트를 위해 사용된다. 만약 서로 다른 권한을 가지는 악의적인 사용자들이 고의




〔그림 2〕 등록 및 발급 프로토콜

적으로 동일한 권한 가명을 선택 및 사용하는 경우에는 공모를 통한 권한 위조가 가능하므로, 권한 가명은 사용자 별로 유일한 값이어야 한다. 이를 위해, AM은 동일한 가명을 사용하는 사용자가 존재하면 다른 가명을 요청한다.

② AM은 사용자의 인증서와 서명을 검증하고 다음의 권한 발급 알고리즘 $A_Issue$을 수행한다.

| $A\_Issue(apk, AUTH) = (\tau_{UN}, (j, \lambda_j))$ |
|---|
| 1. $\tau_{UN} = d_{UN}\pi$ 계산 |
| 2. $(j, \lambda_j) \leftarrow AUTH (1 \le j \le l)$ 선택 |

AM의 공개키 $\pi$를 바탕으로 각 사용자가 랜덤하게 선택한 가명 $d_{UN}$를 곱하여 각 사용자의 권한 연결 값 $\tau_{UN}$를 생성한다. 즉, 권한 연결 값은 IS가 멤버쉽 생성에도 사용하므로 멤버쉽, 익명 사용자, 해당 사용자의 권한을 서로 연결하는 역할을 담당한다. 사용자의 권한 연결 값 $\tau_{UN}$는 사용자가 다른 사용자와 공모하여 권한을 위조하는 것을 방지하기 위해 사용되며, SP가 권한 검증 시, 사용자가 제시한 권한 값이 해당 사용자의 것인지를 확인 할 수 있도록 도와준다. 또한, $\tau_{UN}$는 추후 공개 프로토콜을 통해 익명성이 취소된 사용자의 이전 트랜잭션을 추적하기 위해서도 사용될 수 있다. AM은 총 $l$개의 권한 집합 $AUTH = \{(Q_j, \lambda_j) : Q_j = (j : g_1^{\lambda_j}, AI_j), \lambda_j \in_R \mathbb{Z}_p^*, 1 \le j \le l\}$ 중 사용자에게 적합한 권한 값 $\lambda_j$과 해당 권한 값의 인덱스 $j$를 할당한다. 각 권한 $Q_j$은 특정 권한 검증을 위한 값 $g_1^{\lambda_j}$과 인덱스, 해당 권한의 정보 (AI: Authorization Information)를 포함한다. $\lambda$는 권한 별로 서로 다른 값으로 사용자는 자신이 가진 권한에 따라 서로 다른 여러 개의 $\lambda$를 발급 받는 것이 가능하다. 동일 권한을 가진 사용자들은 서로 동일한 $\lambda$를 가진다. 본 논문에서는 다양한 권한 모델을 적용가능 있도록 권한 모델을 특정한 모델로 국한하지 않고 권한 값은 권한 연결 값과 달리 권한 정보의 인덱스처럼 사용할 수 있는 일반적인 값을 사용한다. 해당 값은 권한에 대한 정보와 연관성을 가지지 않으며 랜덤하게 생성되어, AM이 독립적으로 관리하도록 한다.

③ 사용자가 자신의 권한 값 $\lambda_j$를 안다면, 사용자 간 공모를 통한 자신의 권한 위조가 가능해지므로 사용자에게 숨겨져야만 한다. 또한 가명을 직접 사용자가 IS에게 제시하게 되면 수정 가능하다. 따라서 AM은 사용자의 권한 연결 값과 권한 값을 자신의 개인키로 서명하고 권한 연결 값, 권한 값, 서명, 사용자 UN을 OP의 공개키로 암호화한 $Msg_{AM} = E_{pub_{OP}}[UN||\tau_{UN}||\lambda_j||Sig_{AM}]$를 전달한다.

④ AM은 $I_{AM}^{UN} = [UN : d_{UN}, j, \lambda_j]$를 저장한다.

AM은 SP가 익명 사용자가 가진 권한 값을 검증하여 인가를 할 수 있도록 SP에게 권한 리스트 $AL$ 또한 전달해야만 한다. $AL$은 $AUTH$ 집합의 권한 값 $\lambda$을

제외한 $Q_1,...,Q_i$을 포함한다.

4.3.2.2 멤버 등록

사용자가 실명 인증서를 바탕으로 그룹의 멤버로 등록하는 프로토콜이다.

① 사용자는 실명 인증서와 AM으로부터 받은 $Msg_{AM}$를 바탕으로 OP에게 그룹의 멤버 등록을 요청한다.

② OP는 실명 인증서를 검증하고 $Msg_{AM}$ 값을 검증한다. 검증이 성공하는 경우, OP는 다음의 알고리즘 *Join* 수행을 통해, 사용자에게 멤버 식별자 집합 $ID$에서 유일한 멤버 식별자 $i$를 할당한다. 멤버 식별자를 할당받은 이후부터는 사용자 $UN$을 식별자 $i$를 사용하여 구분한다.

| $Join(ID) = i$ |
|---|
| $i \in_R ID$ 선택 |

③ 사용자의 식별자를 자신의 개인키로 서명한 것과 권한 연결 값 $\tau_i$, 권한 값 $\lambda_j$, $Sig_{AM}$을 IS의 공개키로 암호화한 $Msg_{OP} = E_{pub_{IS}}[i \| Sig_{OP} \| \tau_i \| \lambda_j \| Sig_{AM}]$를 사용자에게 전달한다.

④ OP는 $J_{OP}^i = [i : UN, \tau_i, \lambda_j]$를 저장한다.

4.3.2.3 멤버쉽 발급

그룹의 멤버로 등록된 사용자가 익명 인증을 위해 사용하는 멤버쉽 $(A,x)$과 익명 인가를 위한 $B$를 발급받는 프로토콜로 IS와의 통신을 통해 이루어진다.

① 사용자는 OP로부터 받은 $Msg_{OP}$를 바탕으로 IS에게 멤버쉽 발급을 요청한다.

② IS는 $Msg_{OP}$의 값들을 검증 후, 멤버쉽 발급 알고리즘 $M_Issue$를 수행한다.

멤버쉽 $A_i$는 두 개의 사용자 비밀 값 $x_i, \tau_i$를 포함하고 있으며, 해당 값들은 IS의 비밀 키 $\gamma$와 연결이 되어있으므로 해당 값들을 사용자가 위조하는 것은 불가능하다.

③ IS는 생성된 멤버쉽 비밀키 $gsk[i]$와 익명 인가를 위한 권한 비밀키 $ask[i]$를 사용자에게 전달한다.

| $M\_Issue(gpk, sk_{IS}, MID) = (gsk[i], ask[i])$ |
|---|
| 1. $x_i \in_R \mathbb{Z}_p^*$ 선택 |
| 2. $x_i$와 권한 연결 값 $\tau_i$를 이용하여 멤버쉽 $A_i = g_1^{1/(x_i+\tau_i\gamma)}$를 계산 |
| 3. 생성된 $A_i$와 권한 값 $\lambda_j$를 이용하여 $B_i = A_i^{\lambda_j}$를 생성 |
| 4. 멤버쉽 비밀키를 $gsk[i] = (A_i, x_i, \tau_i)$로 설정하고 권한 비밀키를 $ask[i] = (B_i)$로 설정 |

④ IS는 $Iss_{IS}^i = [i : gsk[i], ask[i]]$를 저장한다.

⑤ 사용자는 IS에게 받은 멤버쉽 비밀키 $gsk[i]$가 올바르게 생성되었는지를 권한 발급 시 선택한 $d_i$에 $apk$를 곱하여 권한 연결 값이 만족하는지를 확인하는 것이 가능하며, $A_i$가 올바르게 생성되었는지 식 (9)가 만족하는지를 검사하여 확인한다.

$$e(A_i, w^{\tau_i} g_2^{x_i}) = e(g_1, g_2) \quad (9)$$

### 4.3.3 익명 인증 및 인가 프로토콜

익명 인증 및 인가 프로토콜은 익명 인가를 위한 메시지 생성, 서명 생성, 익명 인증, 익명 인가 알고리즘을 포함하며, [그림 3]은 해당 프로토콜을 보여준다.

4.3.3.1 익명 인가를 위한 메시지 생성

① 사용자는 다음의 익명 인가를 위한 메시지 생성 알고리즘 $MsgGen$을 통해 $Y_1, Y_2$를 생성한다.

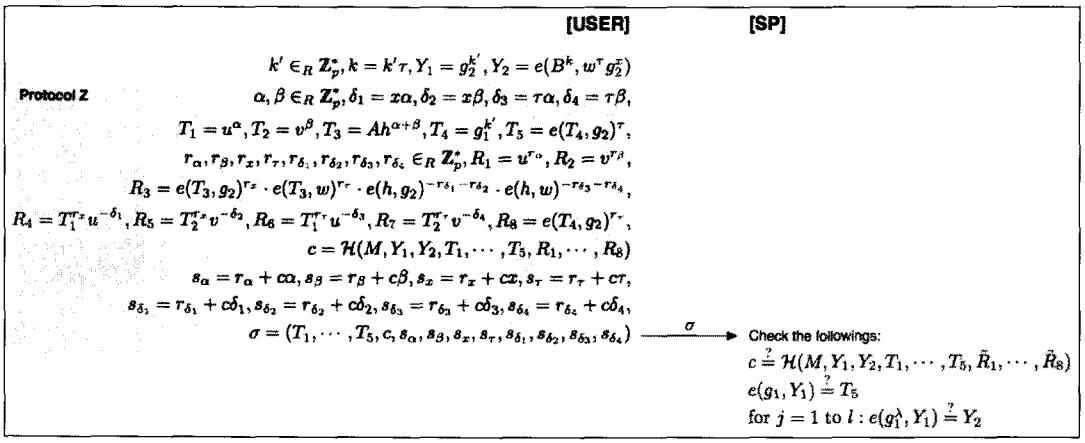
| $MsgGen(gpk, gsk[i], ask[i]) = (Y_1, Y_2)$ |
|---|
| 1. $k' \in_R \mathbb{Z}_p^*$ 선택하여 $k = k'\tau$계산 |
| 2. $Y_1 = g_2^k$, $Y_2 = e(B^k, g_2^x w^\tau)$ 계산 |

② 사용자는 서명을 위해, $Y = (Y_1, Y_2, k')$을 저장한다.

4.3.3.2 익명 인증을 위한 서명

사용자는 익명 인증을 위한 $M \in \{0,1\}^*$에 대한 서명을 생성한다. 해당 서명은 이전에 언급한 것과 같이, 멤버쉽 비밀키 $(A,x)$에 대한 영 지식 증명을 Fiat-Shamir Heuristic을 통해 서명으로 변형한 것이다. 하지만, 짧은 그룹 서명 기법을 약간 변형하여, 멤버쉽 비밀키 $(A,x)$에 대한 영 지식 증명과 함께 권한 연결 값 $\tau$에 대한 영 지식 증명을 추가적으로 수행하며, 익명 인가를 위해, $Y_1, Y_2$를 $c$ 값에 또 다른 메시지처럼 포함시키도록 하며, 서명에는 포함하지 않고 서명과 분리하여 메시지 $Y$를 전송한다. 사용자는 다음의 서명 알고리즘 $Sign$을 수행한다.

| $Sign(gpk, gsk[i], M, Y) = \sigma$ |
|---|
| 1. 프로토콜 Z에 따라 $T_1,..., T_5$를 계산 |
| 2. $Y, T_1,..., T_5, R_1,..., R_8$과 메시지 $M$을 이용하여 $c = H(M, Y, T_1,..., T_5, R_1,..., R_6) \in \mathbb{Z}_p$를 계산 |
| 3. $c$를 사용하여, $s_\alpha, s_\beta, s_x, s_\tau, s_{\delta_1}, s_{\delta_2}, s_{\delta_3}, s_{\delta_4}$를 프로토콜 Z에 따라 계산 |
| 4. 서명 $\sigma = (T_1,.., T_5, c, s_\alpha, s_\beta, s_x, s_\tau, s_{\delta_1}, s_{\delta_2}, s_{\delta_3}, s_{\delta_4})$을 생성 |




〔그림 3〕 익명 인증 및 인가 프로토콜

4.3.3.3 익명 인증

SP는 *Verify* 알고리즘과 사용자로부터 받은 서명 $\sigma$ 검증을 통해 사용자의 서명을 검증하고 정당한 그룹의 멤버임을 인증하는 것이 가능하다. 위 익명 인증을 위한 서명 알고리즘과 마찬가지로, 익명 인증을 위한 서명 검증 시, $c = H(M, Y, T_1, ..., T_5, \widetilde{R}_1, ..., \widetilde{R}_8)$에 인가를 위한 메시지를 포함시켜 계산하고 서명의 $c$ 값과 비교한다.

| $Verify(\sigma, gpk, M) = 1/0$ |
|---|
| 1. 서명의 값과 프로토콜 Z의 식 (1)-(8)을 이용하여 $\widetilde{R}_1, ..., \widetilde{R}_8$, 을 다시 계산<br>2. $c = H(M, Y_1, Y_2, T_1, ..., T_5, \widetilde{R}_1, ..., \widetilde{R}_8)$인지 검사하여 일치하면 1 그렇지 않으면 0 반환 |

4.3.3.4 익명 인가

$Y_1, Y_2$과 AM으로부터 받은 권한 값 리스트 $AL = [Q_1, ..., Q_l]$ $(Q_j = (j : g_1^{\lambda_j} \| AI_j), AI$ : 권한 정보, $1 \le j \le l)$을 통해 사용자의 권한을 익명으로 검증하는 것이 가능하다.

| $Authorization(\sigma, gpk, AL) = j/0$ |
|---|
| 1. $e(g_1, Y_1) = T_5$를 만족하는지를 검사<br>2. 서명에 포함된 $Y_1, Y_2$과 $AL$의 권한 값들을 이용하여 $e(g_1^\lambda, Y_1) = Y_2$이 만족하는지를 검사<br>3. 두 개의 검사가 모두 통과하는 경우에는 서명자의 권한 $j$반환, 그렇지 않으면 0 반환 |

*Authorization* 알고리즘의 첫 번째 단계는 사용자가 다른 사용자와 공모를 통해 권한 값을 위조한 것인지 아닌지 즉, 검증 받는 권한 값이 해당 사용자의 것이 맞는지를 하여 확인하기 위한 과정이다. $T_5$에는 사용자의 권한 연결 값 $\tau$를 포함하고 있으며, 사용자는 익명 인증을 위해 해당 값에 대한 영 지식 증명을 수행한다. 또한 $Y_1, Y_2$는 해당 권한 연결 값을 숨겨서 가지고 있으며, $Y_1$의 $k$가 $k = k'\tau$임을 식 $e(g_1, Y_1) = T_5$로 검증 가능하므로, 사용자는 $\lambda$를 위조하는 것이 불가능하다. 다음과 같은 식의 전개 과정을 통해 권한 확인 가능함을 알 수 있다.

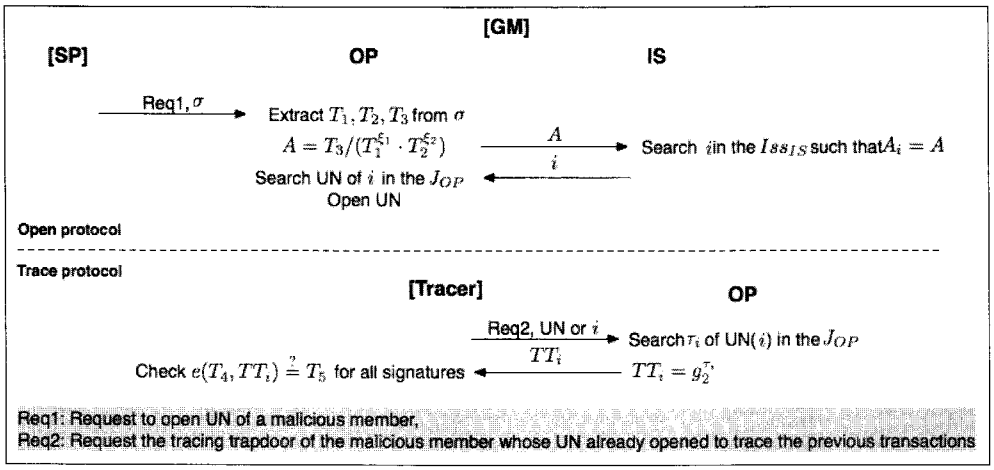
$$Y_2 = e(B^k, g_2^x w^\tau) = e(A^{\lambda k}, g_2^x g_2^{\gamma\tau}) = e(g_1^{\lambda k/x + \tau\gamma}, g_2^{x + \gamma\tau})$$
$$= e(g_1^{\lambda k}, g_2) = e(g_1^\lambda, g_2^k) = e(g_1^\lambda, Y_1),$$
$$T_5 = e(g_1^{k'}, g_2)^\tau = e(g_1, g_2)^{k'\tau} = e(g_1, g_2)^k = e(g_1, Y_1)$$

### 4.3.4 공개 및 추적 프로토콜

공개 및 추적 프로토콜은 공개 알고리즘과 추출 알고리즘, 추적 알고리즘을 포함하며, 〔그림 4〕는 해당 프로토콜을 보여준다.

4.3.4.1 공개

사용자가 익명으로 서비스를 제공받으면서 부정하거나 악의적인 행위를 동작하는 경우, SP는 GM 중 OP에게 해당 사용자의 서명 값을 제시하여 익명성을 취소하고 서명자의 실명 공개를 요청하는 것이 가능하다. GM을 두 개의 기관으로 분리하였으므로 짧은 그룹 서명 기법의 공개 알고리즘을 두 개의 기관이 서로 협력해야만 서명으로부터 서명자의 실명을 공개하는 것이 가능하도록 수정한다. OP는 서명으로부터 멤버쉽 비밀 키의 값인 $A$를 꺼내는 것이 가능하지만, $A$를 사용자의 실명과 연결하는 것은 불가능하다. IS는 멤버쉽 비밀 키와 멤버 식별자만 알고 있고 실명에 대한




〔그림 4〕 공개 및 추적 프로토콜

정보는 가지고 있지 않다. 따라서 두 기관이 협력해야만 완전한 공개 프로토콜이 될 수 있다.

① SP는 특정 서명 $\sigma$를 사용하는 사용자의 익명성 취소를 OP에게 요청한다.

② OP는 SP에게 받은 서명 $\sigma$을 이용하여 다음의 *Open* 알고리즘을 수행한다.

| $Open(\sigma, gpk, sk_{OP}) = A$ |
|---|
| 서명에 포함된 $T_1, T_2, T_3$과 비밀 키 $sk_{OP} = (\xi_1, \xi_2)$를 이용하여 $A = T_3/(T_1^{\xi_1} \cdot T_2^{\xi_2})$를 계산 |

OP는 서명으로부터 멤버쉽 인증서의 값인 $A$를 꺼내는 것이 가능하지만, 사용자의 실명과 연결하는 것은 불가능하므로 IS에게 $A$와 연결된 멤버 식별자를 요청한다.

③ IS는 OP로부터 $A$를 받으면 $A_i = A$인 발급 트랜스크립트 $I_{IS}^i = [A_i : i, x_i, \lambda_j, B_i]$를 찾는다. $I_{IS}^i$에서 멤버 식별자 $i$를 OP에게 전달한다.

④ OP는 IS로부터 받은 멤버 식별자 $i$를 가지는 사용자의 등록 트랜스크립트 $J_{OP}^i = [i : UN, \lambda_j]$를 찾는다. OP는 $J_{OP}^i$에서 서명자의 실명 UN을 찾아 공개한다.

4.3.4.2 추적

추적 프로토콜은 짧은 그룹 서명 기법에서는 포함하고 있지 않지만, 본 논문의 익명 인증 및 인가 기법에서는 사용자의 권한 값 위조를 막기 위한 권한 연결 값 $\tau$을 사용자의 tracing trapdoor $TT = g_2^{\tau}$에 사용하여 익명성이 취소된 사용자의 이전 트랜잭션 추적이 가능하도록 한다. 또한 기존의 추적 가능 서명 기법의 추적 프로토콜과 달리 권한 값을 이용하여 사용자가 이전 트랜잭션에서 권한 값을 위조하여 사용했는지 아닌지의 여부를 다시 한 번 검증하는 것도 가능하다.

① 추적자(tracer) 혹은 추적 에이전트(tracing agent)는 익명성이 취소된 사용자 $i$의 이전 트랜잭션 추적을 위한 tracing trapdoor를 OP에게 요청한다.

② OP는 다음의 *Reveal* 알고리즘을 수행하고, 수행 결과 tracing trapdoor를 추적자 혹은 추적 에이전트에게 전달한다.

| $Reveal(i, J_{OP}^i) = TT_i$ |
|---|
| 1. 사용자 $i$의 등록 프로토콜 트랜스크립트 $J_{OP}^i$에서 $\tau_i$를 찾음<br>2. tracing trapdoor $TT_i = g_2^{\tau_i}$를 계산 |

③ OP로부터 $TT$를 받은 후, *Trace* 알고리즘을 수행한다.

| $Trace(\sigma, TT, gpk) = 1/0$ |
|---|
| 1. 서명들의 $T_5$값이 $e(T_4, g_2^{\tau_i})$와 같은지 검사<br>2. 검사가 통과하는 경우에는 1 반환, 그렇지 않으면 0 반환 |

만약, 사용자의 이전 트랜잭션의 권한 값을 다시 검증하기를 원하는 경우에는 OP에게 $TT$ 값 외에 권한 값 $\lambda_j$를 요청한다. 해당 값을 받아 $Y_2^{-\lambda_j} = T_5$인지 검사하여 확인하는 것이 가능하다.

### 4.4 익명 인증 및 인가 기법의 안전성

제안한 익명 인증 및 인가를 위한 기법의 안전함을 보이기 위해서는 먼저, SDH 표현에 대한 영지식 프로토콜인 프로토콜 Z가 honest-verifier 영지식 증명(zero-knowledge proof of knowledge)임을 보여야 한다. 익명 인증 및 인가를 제공하면서 3.4절에서 언급한 그룹 서명 기법의 안전성 요구사항인 정당성, 충분 익명성, 충분 추적성을 만족해야만 한다.

**정의 1. (프로토콜 Z의 honest-verifier 영지식 증명)** 프로토콜 Z가 명백하고 정직한 검증자에 대해 시뮬레이터와 추출자(extractor)가 존재하면, 프로토콜 Z는 honest-verifier 영지식 증명이다.

**정의 2. (익명 인증 및 인가 기법의 정당성)** 만약 다음의 네 가지 조건을 만족하면, 보안 파라미터 $k$를 가지는 익명 인증 및 인가 기법은 정당 (correct)하다.

- 서명 정당성(Sign-correctness): 모든 $m$에 대해, $Verify(m, Sign_U(m)) = 1$.
- 인가 정당성: 임의의 $U$에 대해, $Authorization(Sign_U(m), gpk, AL) = j$.
- 공개 정당성(Open-correctness): 임의의 $m$에 대해, $Open(Sign_U(m), sk_{OP}) = U$.
- 추적 정당성(Trace-correctness): 임의의 $m$에 대해, $Trace(Sign_U(m), Reveal(U), gpk) = 1$.

익명 인증 및 인가 기법의 서명은 사용자를 식별 가능한 멤버쉽 $A$를 포함한 짧은 그룹 서명의 동일한 서명 값 $T_1, T_2, T_3$과 부가적으로 사용자를 식별 가능한 또 다른 비밀 값인 $\tau$ 가 포함된 $T_5, Y_1$, 두 개의 식별 가능한 값을 이용하여 생성하는 $Y_2$를 포함한다. 따라서 익명 인증 및 인가 기법의 충분 익명성을 증명하기 위해서 서명에 포함된 값들로부터 $A, \tau$를 꺼낼 수 없음을 증명해야 한다.

**정의 3. (익명 인증 및 인가 기법의 충분-익명성)** 만약 익명 인증 및 인가 기법이 다음의 조건을 만족한다면, 익명 인증 및 인가 기법은 충분 익명성(fully-anonymous)을 가진다.

- 익명 인증 및 인가 기법의 서명 중 $T_1, T_2, T_3$로부터 사용자의 식별자인 $A$를 알아낼 수 없음
- 익명 인증 및 인가 기법의 서명 중 $T_5$으로부터 사용자의 식별자인 $\tau$를 알아낼 수 없음
- 익명 인증 및 인가 기법의 인가 메시지 $Y_1, Y_2$로부터 사용자의 식별자인 멤버쉽 비밀 키 $(A, \tau, x)$를 알아낼 수 없음

**정의 4. (익명 인증 및 인가 기법의 충분-추적성)** 만약 주어진 서명에 대해 공개 프로토콜을 수행 했을 때, 서명자가 그룹의 멤버 중 하나 일 때, 익명 인증 및 인가 기법은 충분 추적성(full-traceable)을 가진다.

### 4.5 멤버쉽 취소

짧은 그룹 서명 기법은 공개 프로토콜을 통해 악의적인 행위를 한 사용자의 익명성을 취소하는 것이 가능하며, 더 이상 발급받은 멤버쉽 비밀 키로 익명 인증을 위한 서명을 생성하지 못하도록 멤버쉽을 취소(revocation)하는 것이 가능하다. 즉, 사용자를 그룹으로부터 제외시키는 것을 말하며, 제외된 사용자는 더 이상 멤버쉽 비밀 키를 사용할 수 없도록 하고 나머지 사용자들은 서명 가능하도록 그룹의 공개키와 나머지 사용자의 멤버쉽 비밀 키를 업데이트해야 한다. 이를 위해, 짧은 그룹 서명 기법에서는 취소 기법을 제공한다. 먼저 취소 기관(RA: Revocation Authority)은 멤버쉽이 취소된 모든 사용자 $1,...,r$의 멤버쉽 비밀 키와 취소되지 않은 사용자의 권한 값 업데이트를 위한 익명 인가 값들을 포함하는 취소 목록 RL (revocation list) 을 배포한다.

$$RL = \begin{bmatrix} \{A_1^*, x_1, d_1, (B_1^*(1), ..., B_1^*(l)\}, ..., \\ \{A_r^*, x_r, d_r, (B_r^*(1), ..., B_r^*(l)\} \end{bmatrix}$$

$$(A_i^* = g_2^{1/(\gamma + x_i)} \in G_2, A_i = \psi(A_i^*), B_i^*(j) = (A_i^*)^{\lambda_j} \in G_2)$$

RL이 주어지면 취소되지 않은 모든 멤버는 독립적으로 그룹 공개키와 자신의 멤버쉽 비밀키를 업데이트 하는 것이 가능하다. 먼저, RL의 하나의 취소 사용자의 멤버쉽을 이용한 방법을 보이고, 동일 과정을 RL에 포함된 사용자의 수($r$)만큼 반복하면 RL의 모든 취소된 멤버쉽을 그룹 공개키 및 자신의 멤버쉽 비밀키 업데이트에 적용하는 것이 가능하다. RL의 취소된 첫 번째 멤버쉽 비밀키를 그룹 공개키를 업데이트 하는 방법은 다음과 같다.

$$\hat{g_1} \leftarrow \psi(A_1^*),\ \hat{g_2} \leftarrow A_1^*,$$

$$\hat{w} \leftarrow (g_2 \cdot (A_1^*)^{-x_1})^{1/\tau_1} = g^{\gamma/(x_1 + \gamma\tau_1)} = (\hat{g_2})^{\gamma}$$

그룹의 공개키가 업데이트 되었으므로, 각 취소되지 않은 사용자는 추후 자신의 멤버쉽 비밀 키로 익명 인증 및 인가를 받기 위해 멤버쉽 비밀키 $A = g_1^{1/x+\gamma r}$를 $(\hat{A})^{x+\gamma r} = \hat{g_1}$를 만족하도록 다음과 같이 업데이트

해야할 필요가 있다.

$$\hat{A} \leftarrow \psi(A_1^*)^{\frac{d_1}{d_1x-dx_1}} / A^{\frac{d}{d_1x-dx_1}}$$
$$= g_1^{\frac{d_1}{(x_1+\gamma\tau_1)(d_1x-dx_1)}} / g_1^{\frac{d}{(x+\gamma\tau)(d_1x-dx_1)}}$$
$$= g_1^{\frac{d_1x+dd_1\gamma\pi-dx_1+dd_1\gamma\pi}{(d_1x-dx_1)(x_1+\gamma\tau_1)(x+\gamma\tau)}} = g_1^{\frac{1}{(x_1+\gamma\tau_1)(x+\gamma\tau)}} = (\hat{g_1})^{\frac{1}{x+\gamma\tau}}$$

익명 인가를 위해 발급받은 권한 값 $B$는 $A^{\lambda}$로 구성되어 있으므로, 멤버쉽 비밀 키 $A$가 $\hat{A}$로 업데이트 되면, $B$ 또한 $\hat{B}=(\hat{B})^{\lambda}$로 업데이트를 해야 익명 인가를 정상적으로 받을 수 있게 된다. 사용자는 RL의 자신과 동일한 권한 인덱스를 가진 값 $B_1^*(j)$과 자신의 $B$를 이용하여 다음과 같이 $\hat{A}$를 업데이트 하는 방식과 유사하게 계산할 수 있다.

$$\hat{B} \leftarrow \psi(B_1^*)^{\frac{d_1}{d_1x-dx_1}} / B^{\frac{d}{d_1x-dx_1}}$$
$$= g_1^{\frac{\lambda d_1}{(x_1+\gamma\tau_1)(d_1x-dx_1)}} / g_1^{\frac{\lambda d}{(x+\gamma\tau)(d_1x-dx_1)}}$$
$$= g_1^{\frac{\lambda d_1(x+\gamma d\pi)-\lambda d(x_1+\gamma d_1\pi)}{(d_1x-dx_1)(x_1+\gamma\tau_1)(x+\gamma\tau)}} = (\hat{g_1})^{\frac{\lambda}{x+\gamma\tau}} = (\hat{A})^{\lambda}$$

익명 인가를 위해 메시지를 생성하는 사용자뿐만 아니라 익명 인가를 수행해야하는 SP 또한 그룹 공개키가 업데이트되므로 $AL$에 포함된 각 $Q_j$을 $\hat{Q_j}$로 업데이트해야 한다. 이를 위해, SP는 취소된 사용자가 하나인 경우 혹은 첫 번째 취소된 사용자의 멤버쉽을 이용하는 경우에는 RL의 $B_1^*(j)$ 리스트를 그대로 이용하여 $g_1^{\lambda_j}$를 $(\hat{g_1})^{\lambda_j}$로 다음과 같이 간단히 업데이트 할 수 있다.

$$(\hat{g_1})^{\lambda_j} = \psi(B_1^*(j)) = g_1^{\lambda_j/x_1+\gamma\tau_1}$$

RL의 나머지 취소된 멤버쉽을 적용해야 하는 경우에는 사용자가 $\hat{B}$를 업데이트 하는 방식과 동일하게 업데이트 할 수 있다. 다음 식의 $(\hat{g_1'}) = g^{1/(x_1+\gamma\tau_1)(x_2+\gamma\tau_2)}$로 RL의 두 번째 멤버쉽이 적용되어 업데이트된 공개키이다.

$$(\hat{g_1'})^{\lambda_j} = \psi(B_2^*)^{\frac{d_2}{(d_2x_1-d_1x_2)}} / \psi(B_1^*)^{\frac{d_1}{(d_2x_1-d_1x_2)}}$$
$$= g_1^{\frac{\lambda_j d_2}{(x_2+\gamma\tau_2)(d_2x_1-d_1x_2)}} / g_1^{\frac{\lambda_j d_1}{(x_1+\gamma\tau_1)(d_2x_1-d_1x_2)}}$$
$$= g_1^{\frac{\lambda_j d_2(x_1+\gamma d_1\pi)-\lambda_j d_1(x_2+\gamma d_2\pi)}{(d_2x_1-d_1x_2)(x_2+\gamma\tau_2)(x_1+\gamma\tau_1)}}$$
$$= g_1^{\frac{\lambda_j d_2x_1-\lambda_j d_1x_2+\lambda_j\gamma d_1d_2\pi-\lambda_j\gamma d_1d_2\pi}{(d_2x_1-d_1x_2)(x_2+\gamma\tau_2)(x_1+\gamma\tau_1)}}$$
$$= g_1^{\frac{\lambda_j}{(x_2+\gamma\tau_2)(x_1+\gamma\tau_1)}} = (\hat{g_1})^{\frac{\lambda_j}{x_2+\gamma\tau_2}}$$

## V. 분 석

제안한 익명 인증 및 인가 기법이 4.4절에 정의된 안전성 요구사항을 모두 만족함을 증명하고, 연산량 및 서명 길이를 통해 성능을 비교한다. 또한 익명 인가를 위한 사용자 권한 값에 대해 논한다.

### 5.1 안전성 분석

먼저, 프로토콜 Z가 honest-verifier 영지식 증명임을 보이고, 익명 인증 및 인가 기법이 그룹 서명의 안전성 요구사항인 정당성, 충분 익명성, 충분 추적성을 만족함을 보인다. 충분 추적성의 정의는 공개 프로토콜을 수행했을 때, 서명으로부터 해당 서명을 한 그룹 멤버를 찾을 수 있음을 의미한다. 서명으로부터 익명 서명자를 찾을 수 있음은 서명이 위조 불가능하여 반드시 서명의 서명자를 찾아 낼 수 있음을 의미하므로 서명이 위조 불가능함을 보여 이를 증명한다.

#### 5.1.1 프로토콜 Z의 안전성

정리 1. 프로토콜 Z는 DLDH 가정에 기반하여 SDH 표현에 대한 honest-verifier 영지식 증명이다.

증명. 다음의 세 개의 보조정리에 따라 정리 1이 만족한다.

보조정리 1.1. [completeness] 프로토콜 Z는 명백하다.

증명. 만약, 앨리스가 SDH 표현 $(A,x,\tau)$를 가진 정직한 증명자라면, 앨리스는 프로토콜 Z를 따라 계산한다. 이러한 경우, $u^{s_\alpha} = u^{r_\alpha + c\alpha} = (u^\alpha)^c \cdot u^{r_\alpha}$ $= T_1^c \cdot R_1$ 이므로 식 (1)을 만족하며, 유사한 방식으로 식 (2)도 만족한다. $T_1^{s_{\delta_1}} u^{-s_{\delta_1}} = (u^\alpha)^{r_x+cx} u^{-r_{\delta_1}-cx\alpha}$ $= (u^\alpha)^{r_x} u^{-r_{\delta_1}} = T_1^{r_x} u^{-r_{\delta_1}} = R_4$ 이므로, 식 (4)을 만족하며, 유사한 방식으로 식 (5)-(7)도 만족한다. 또한, $e(T_4,g_2)^{s_\tau} = e(T_4,g_2)^{r_\tau + c\tau} = (e(T_4,g_2)^\tau)^c \cdot e(T_4,g_2)^{r_\tau}$ $= T_5^c \cdot R_6$ 이므로, 식 (8)을 만족한다. 마지막으로,

$$e(T_3,g_2)^{s_x} \cdot e(T_3,w)^{s_\tau} \cdot e(h,g_2)^{-s_{\delta_1}-s_{\delta_2}} \cdot e(h,w)^{-s_{\delta_1}-s_{\delta_2}}$$
$$= e(T_3,g_2)^{r_x+cx} \cdot e(h,w)^{-(r_\tau+c\tau)(r_\alpha+c\alpha+r_\beta+c\beta)}$$
$$\cdot e(h,g_2)^{-r_{\delta_1}-r_{\delta_2}-cx\alpha-cx\beta}$$

$$= e(T_3, g^{x_2})^c \cdot e(h^{-\alpha-\beta}, w^{\tau} g^{x_2})^c$$
$$\cdot (e(T_3, g_2)^{r_x} \cdot e(h,w)^{-r_\tau(r_\alpha + r_\beta)} \cdot e(h, g_2)^{-r_{\delta_1} - r_{\delta_2}})$$
$$= e(T_3 h^{-\alpha-\beta}, w^{\tau} g_2^{x})^c \cdot e(T_3, w^{\tau})^{-c} \cdot R_3$$
$$= (e(g_1, g_2)/e(T_3, w^{\tau}))^c \cdot R_3$$

이므로 식 (3)을 만족한다.

보조정리 1.2. [Simulator] 정직한 검증자에 대해, DLDH 가정에 기반하여 프로토콜 Z의 트랜스크립트에 대한 시뮬레이터가 존재한다.

증명. 먼저, 프로토콜 Z의 트랜스크립트를 출력하는 시뮬레이터를 묘사한다. 시뮬레이터는 $A \xleftarrow{R} G_1$, $\alpha, \beta, \tau, k' \xleftarrow{R} \mathbb{Z}_p$를 선택하고, $T_1 \leftarrow u^{\alpha}$, $T_2 \leftarrow v^{\beta}$, $T_3 \leftarrow A h^{\alpha+\beta}$, $T_4 \leftarrow g_1^{k'}$, $T_5 \leftarrow e(T_4, g_2)^{\tau}$를 계산한다. 시뮬레이터는 $c \xleftarrow{R} \mathbb{Z}_p$와 $s_\alpha, s_\beta, s_x, s_\tau, s_{\delta_1}, s_{\delta_2}, s_{\delta_3}, s_{\delta_4} \xleftarrow{R} \mathbb{Z}_p$를 선택하고, 프로토콜 Z와 같이 $R_1, \cdots, R_8$을 계산한다. 이 결과 값들은 식 (1)-(8)을 만족하고, 값 $R_1, \cdots, R_8$은 실제 트랜스크립트와 같이 분산된 값이다. 시뮬레이터는 결과적으로 트랜스크립트 $(T_1, \cdots, T_5, R_1, \cdots, R_8, c, s_\alpha, s_\beta, s_x, s_\tau, s_{\delta_1}, s_{\delta_2}, s_{\delta_3}, s_{\delta_4})$를 출력한다. 이 트랜스크립트는 DLDH 가정을 가정할 때, 프로토콜 Z의 트랜스크립트와 구별 불가능하다.

보조정리 1.3. [추출자] 프로토콜 Z에 대한 추출자가 존재한다.

증명. 프로토콜 Z의 증명자가 도전 값 $c$를 받기 직전의 순간으로 증명자를 다시 돌릴 수 있는 추출자가 있다고 가정하자. 그러면 추출자는 다음의 두 개의 프로토콜 트랜스크립트를 얻을 수 있다.

$$(T_1, \cdots, T_5, R_1, \cdots, R_8, c, s_\alpha, s_\beta, s_x, s_\tau, s_{\delta_1}, s_{\delta_2}, s_{\delta_3}, s_{\delta_4})$$
$$(T_1, \cdots, T_5, R_1, \cdots, R_8, c', s'_\alpha, s'_\beta, s'_x, s'_\tau, s'_{\delta_1}, s'_{\delta_2}, s'_{\delta_3}, s'_{\delta_4})$$

간결하게 하기 위해, $\Delta c = c - c'$, $\Delta s_\alpha = s_\alpha - s'_\alpha$라고 하고, $\Delta s_\beta, \Delta s_{\delta_1}, \Delta s_{\delta_2}, \Delta s_{\delta_3}, \Delta s_{\delta_4}$로 마찬가지로 가정한다. 먼저, 식 (1)의 두 개의 인스턴스를 나누면, $u^{\Delta s_\alpha} = T_1^{\Delta c}$을 얻을 수 있다. $\tilde{\alpha} = \Delta s_\alpha / \Delta c$를 루트로 하면, $u^{\tilde{\alpha}} = T_1$가 되고, 마찬가지로, 식 (2)로부터 $v^{\tilde{\beta}} = T_2$인 $\tilde{\beta} = \Delta s_\beta / \Delta c$를 얻을 수 있다. 식(8)의 두 개의 인스턴스를 나누면, $e(T_4, g_2)^{\Delta s_\tau} = T_5^{\Delta c}$이 되며, $\tilde{\tau} = \Delta s_\tau / \Delta c$를 얻을 수 있다. 식 (4)의 두 개의 인스턴스를 나누어 얻은 $T_1^{\Delta s_x} = u^{\Delta s_{\delta_1}}$에서 $T_1 = u^{\tilde{\alpha}}$을 대체하면 $u^{\tilde{\alpha}\Delta s_x} = u^{\Delta s_{\delta_1}}$ 혹은 $\Delta s_{\delta_1} = \tilde{\alpha} \Delta s_x$를 얻을 수 있다. 마찬가지로 식 (5)-(7)로부터, $\Delta s_{\delta_2} = \tilde{\beta} \Delta s_x$, $\Delta s_{\delta_3} = \tilde{\alpha} \Delta s_\tau$, $\Delta s_{\delta_4} = \tilde{\beta} \Delta s_\tau$를 얻을 수 있다. 마지막으로, 식 (3)의 두 개의 인스턴스를 나누면, 다음의 식을 얻을 수 있다.

$$e(g_1, g_2)^{\Delta c}$$
$$= e(T_3, g_2)^{\Delta s_x} \cdot e(T_3, w)^{\Delta s_\tau} \cdot e(h, g_2)^{-\Delta s_{\delta_1} - \Delta s_{\delta_2}}$$
$$\cdot e(h, w)^{-\Delta s_{\delta_3} - \Delta s_{\delta_4}}$$
$$= e(T_3, g_2)^{\Delta s_x} \cdot e(T_3, w)^{\Delta s_\tau} \cdot e(h, g_2)^{-\tilde{\alpha}\Delta s_x - \tilde{\beta}\Delta s_x}$$
$$\cdot e(h, w)^{-\tilde{\alpha}\Delta s_\tau - \tilde{\beta}\Delta s_\tau}$$

$\Delta c$로 지수부를 모두 나누고, $\tilde{x} = \Delta s_x / \Delta c$라고 두면, 식 $e(g_1, g_2) = e(T_3, g_2)^{\tilde{x}} \cdot e(T_3, w)^{\tilde{\tau}} \cdot e(h, g_2)^{-\tilde{x}(\tilde{\alpha}+\tilde{\beta})} \cdot e(h, w)^{-\tilde{\tau}(\tilde{\alpha}+\tilde{\beta})}$을 얻을 수 있고, 다시 식 $e(g_1, g_2) = e(T_3 h^{-\tilde{\alpha}-\tilde{\beta}}, g_2^{\tilde{x}} w^{\tilde{\tau}})$로 정리할 수 있다. $\tilde{A} = T_3 h^{-\tilde{\alpha}-\tilde{\beta}}$라고 하면, $e(g_1, g_2) = e(\tilde{A}, g_2^{\tilde{x}} w^{\tilde{\tau}})$가 된다. 따라서 추출자는 SDH 표현 $(\tilde{A}, \tilde{x}, \tilde{\tau})$를 얻을 수 있다.

### 5.1.2 익명 인증 및 인가 기법의 안전성

정리 2. 4.2절의 익명 인증 및 인가 기법은 정당(correct)하다.

정리 3. 선형 암호화 기법이 의미론적으로 안전함과, $\mathbb{G}_2$에서의 이산대수 문제가 실행 불가능함, 겹선형 쌍함수의 역연산이 불가능함에 의해, 익명 인증 및 인가 기법은 fully-anonymous하다.

보조정리 3.1. 선형 암호화 기법이 $\mathbb{G}_1$에서 $(t', \epsilon')$의 미론적으로 안전하다면, 익명 인증 및 인가 기법의 서명 값 중 $T_1, T_2, T_3$로부터 사용자의 식별자를 알아낼 수 없다.

이는 짧은 그룹 서명 기법의 충분 익명성 증명과 동일하게 진행가능 하므로 본 논문에서는 생략한다.

보조정리 3.2. 만약 $\mathbb{G}_2$에서의 이산대수 문제가 실행 불가능하다면, $Y_1$메시지로부터 $r$를 꺼내는 것은 불가능하다.

증명. 익명 인증 및 인가 기법의 서명 값 중

$Y_1 = g_2^k = g_2^{k'\tau}$에서 $k$를 다항식 시간 안에 알아낼 수 있는 알고리즘 A를 가정하고, 알고리즘 A를 이용하여 $\mathbb{G}_2$에서의 이산대수 문제를 풀려는 알고리즘 B를 가정하자. 알고리즘 B는 확인자 C로부터 주어진 입력 $(g_2, g_2^\tau)$과 자신이 랜덤하게 선택한 $k' \in_R \mathbb{Z}_p^*$를 이용하여 $Y_1 = (g_2^\tau)^{k'}$를 생성하고 $g_2, Y_1$을 알고리즘 A에게 넘겨준다. 알고리즘 A는 다항식 시간 안에 $k$를 계산할 수 있고, 알고리즘 B는 이로부터 이산대수 문제의 $\tau = k/k'$를 알아낼 수 있다.

보조정리 3.3 만약 $\mathbb{G}_2$에서의 이산대수 문제가 실행 불가능하고, 겹선형 쌍함수의 역연산이 불가능하다면, 서명 값 $T_5$로부터 $\tau$를 꺼내는 것은 불가능하다.

증명. 익명 인증 및 인가 기법의 서명 값 중 $T_5 = e(g_1^{k'}, g_2)^\tau = e(g_1^{k'}, g_2^\tau)$에서 $g_2^\tau$를 다항식 시간 안에 알아낼 수 있는 알고리즘 A를 가정하고, 알고리즘 A를 이용하여 겹선형 쌍함수의 역연산 하려는 알고리즘 B를 가정하자. 알고리즘 B는 확인자 C로부터 주어진 입력 $(g_1, g_2, e(g_1, g_2^\tau))$과 자신이 랜덤하게 선택한 $k' \in_R \mathbb{Z}_p^*$를 이용하여, $T_4 = g_1^{k'}$과 $T_5 = e(g_1, g_2^\tau)^{k'}$을 계산하여 $g_1, g_2, T_4, T_5$를 알고리즘 A에게 넘겨준다. 알고리즘 A는 다항식 시간 안에 $g_2^\tau$를 계산할 수 있고, 알고리즘 B는 이로부터 $e(g_1, g_2^\tau)$의 겹선형 쌍함수의 역연산 값을 알아낼 수 있다. $g_2^\tau$에서 $\tau$를 꺼내는 문제는 위의 보조정리 3.2에 의해서 불가능함을 보였다.

보조정리 3.4 만약 $\mathbb{G}_2$에서의 이산대수 문제가 실행 불가능하고, 겹선형 쌍함수의 역연산이 불가능하다면, 서명 값 $Y_2$로부터 멤버쉽 비밀 키 각각의 값 $(A, \tau, x)$를 꺼내는 것은 불가능하다.

이는 보조정리 3.2와 보조정리 3.3과 동일한 방법으로 증명가능하다. 또한, $Y_2$의 경우에는 사용자가 계산을 위해서는 멤버쉽 비밀 키 $gsk[i] = (A, \tau, x)$를 이용하지만, 계산 결과 값은 해당 비밀 키 값들이 모두 상쇄되므로 $Y_2 = e(g_1^{\tau\lambda}, g_2)$로부터 $(A, \tau, x)$를 꺼내는 것은 불가능하므로 증명을 생략한다.

정리 4. 만약 SDH가 $\mathbb{G}_1, \mathbb{G}_2$에서 $(q, t', \epsilon')$ 어렵다면, 익명 인증 및 인가 기법은 $(t, q_H, q_S, n, \epsilon)$-fully traceable하다. 여기서, $n = q-1, \epsilon = 4n\sqrt{2\epsilon' q_H} + n/p$, $t = \Theta(1) \cdot t'$이며, $q_H$는 공격자의 해쉬 함수 쿼리 수, $q_S$는 공격자의 서명 쿼리 수이고, $n$은 그룹의 멤버 수를 의미한다.

증명. 짧은 그룹 서명 기법의 경우에는 [3]의 SDH 인스턴스를 $(A, x)$로 변화시키고, Forking Lemma를 이용하는 서명 위조자를 통해 충분 추적성을 증명하였다. 따라서 $n+2$개의 SDH 인스턴스를 $m(m<n)$개의 익명 인증 및 인가 기법의 멤버쉽 비밀 키인 $(A_i, x_i, \tau_i)$로 변화시키는 것이 가능하다면, 짧은 그룹 서명 기법과 동일한 형태로 증명이 가능하다. 물론, 오라클에서 공격자 알고리즘에게 제공하는 값은 익명 인증 및 인가 기법의 형태에 따라 약간 변형되지만 전체적인 증명 방식은 동일하다. 또한 멤버쉽 비밀 키 중 권한 연결 값은 원래 사용자마다 서로 다른 값을 가지지만, SDH 인스턴스를 멤버쉽 비밀 키의 형태로 변화시키기 위해서 동일한 값이어야만 가능하다. 하지만, 권한 연결 값은 사용자가 선택하는 가명에 의해 결정되므로 동일 값으로 고정하여도 다른 멤버쉽 비밀 키를 통해 서명자를 유일하게 식별하는 것이 가능하므로, 모든 멤버쉽 비밀 키는 타당한 서명키 값이라 할 수 있다. 그러므로 본 논문에서는 $n+2$개의 SDH 인스턴스를 $m$개의 익명 인증 및 인가 기법의 멤버쉽 비밀 키인 $(A_i, x_i, \tau_i)$로 변화시키는 것을 보이고, 이 후 증명은 짧은 그룹 서명 기법과 동일한 절차를 통해 이루어질 수 있다. $n+2$개의 SDH 인스턴스 $(g_1', g_2', (g_2')^\gamma, (g_2')^{\gamma^2}, \cdots, (g_2')^{\gamma^n})$가 주어졌다고 가정한다. $x_1, \cdots, x_m, \tau \in \mathbb{Z}_p^*$를 선택하고, 모든 $i(1 \le i \le m)$에 대해 $y_i = x_i/\tau$를 계산한다.

$$f(\gamma) = \prod_{i=1}^{n-1}(\gamma + y_i) = \sum_{i=0}^{q-1}\alpha_i\gamma^i,\ f(\gamma)_i = f(\gamma)/(\gamma + y_i)$$
$$= \prod_{j=1, j\neq i}^{n-1}(\gamma + y_i) = \sum_{i=0}^{q-2}\beta_j\gamma^j$$

위의 다항식 정의를 사용하여 [3]와 유사한 방법을 사용한다.

$$g_2 \leftarrow \prod_{i=0}^{n-1}(g_2')^{\gamma^i\alpha_i} = (g_2')^{f(\gamma)},\ g_1 \leftarrow \Psi(g_2),$$
$$w \leftarrow \prod_{i=0}^{n}(g_2')^{\gamma^i\alpha_{i-1}} = (g_2')^{\gamma f(\gamma)} = g^{\gamma_2},$$
$$A_i^* \leftarrow \prod_{j=0}^{n-2}(g_2')^{\frac{1}{\tau}\gamma^j\beta_j} = (g_2')^{\frac{1}{\tau}f_i(\gamma)} = g_2^{\frac{1}{\tau}(\frac{1}{\gamma+y_i})} = g_2^{\frac{1}{x_i+\tau\gamma}},$$
$$A_i \leftarrow \Psi(A_i^*)$$

그룹 공개키인 $g_1, g_2, w$와 멤버쉽 비밀 키 $(A_i, x_i, \tau_i)$를 생성하는 것이 가능하다.

### 5.2 성능 분석

제안한 익명 인증 및 인가 기법은 짧은 그룹 서명 생성 시, commit를 위한 $c$ 값을 계산 시, 익명 인가를 위한 메시지를 포함하고 해당 메시지를 추가로 전송하는 부분에 차이가 있으며, 멤버쉽 생성 시, 권한 연결 값이 추가되므로 짧은 그룹 서명 기법보다 긴 길이의 서명을 가진다. 사용자가 가지는 권한의 수가 증가하는 경우에는 서명 요소와 길이가 수에 따라 비례하여 증가할 수도 있다. 하지만 이는 적용하는 권한 모델에서의 권한 값을 조정하여 해결하는 것이 가능하다. 예를 들어, 역할 기반 권한 모델의 경우에는 선형적인 역할 기반이 아닌 계층적인 역할 기반 접근 제어 모델을 사용하면 특정 권한을 위해 필요한 메시지의 수가 감소하도록 할 수 있다.

연산량의 경우에는 지수연산과 쌍함수 연산을 고려했을 때, 실제로 메시지 $Y_1, Y_2$를 계산하기 위해서는 4번의 지수 연산과 1번의 쌍함수 연산이 필요하지만, 이는 모두 사전 계산하는 것이 가능하다. 짧은 그룹 서명 기법과 마찬가지로 $e(g_1,g_2)$는 사전 계산하여 미리 저장해두었다고 가정하면, 익명 인증 및 인가 기법에서는 SP가 서명 검증을 위해서 추가적인 지수 연산을 필요로 하지만, 쌍함수 연산은 한 번 더 적게 수행하면 된다. SP가 사용자의 권한을 검증하는 경우에는 쌍함수 연산만 필요하며 하나의 권한을 확인하기 위해서는 총 권한의 수만큼 쌍함수 연산이 필요하다. 이는 사용자가 서명 생성 시 마다 랜덤한 $r$을 선택하고 동일 권한에 대해 매번 서로 다른 메시지를 생성하여 서명 사이의 연결성(linkability)를 제거하가 위한 것이다. [표 2]는 짧은 그룹 서명 기법(BBS04)과 익명 인증 및 인가 기법(AAA)의 익명 인가를 위한 메시지를 포함한 서명의 요소, 길이와 연산량을 비교한 것이다. 여기서, e는 지수 연산을 의미하며, p는 쌍함수 연산을 의미한다. $l$은 권한의 개수를 의미한다.

### 5.3 사용자 권한 값에 대한 고찰

일반적인 비즈니스 모델에서는 다중 사용자가 동일 권한을 가질 수 있으며, 제안하는 기법은 동일 권한을 가진 사용자에게 동일 랜덤 값을 할당함으로써 이를 해결할 수 있다. 동일 권한을 가지는 사용자는 동일 권한 값을 가지므로 권한 값을 통해 사용자를 구분하는 것은 불가능하다. 권한 값을 랜덤 값으로 사용하므로 다양한 접근 제어 모델에 적용하는 것이 가능하다.

[표 2] 서명 길이와 연산량 비교

| | | | BBS04 | AAA |
|---|---|---|---|---|
| 서명길이 | | 요소 단위 | 9 | 16 |
| | | 비트 단위 | 1533 | 4433 |
| 연산 | 인증 | 서명 생성 | 12 e, 2 p | 19 e, 2 p |
| | | 서명 검증 | 12 e, 3 p | 19 e, 2 p |
| | 인가 | 인가메시지생성 | x | 4 e, 1 p |
| | | 익명 인가 | x | $2l$ p |

단일 사용자는 여러 개의 권한을 가지는 것이 가능하며, 권한 수정 또한 가능하다. 제안한 기법은 단일 사용자가 하나의 권한을 가지는 경우에 해당하므로 여러 개의 권한을 가지는 경우와 권한의 수정이 필요한 경우를 처리 가능함을 보인다.

단일 사용자가 여러 개의 권한을 가지는 경우에는 사용자가 AM으로부터 여러 개의 권한 값 $\lambda_j,...,\lambda_k$을 할당받아 OP에게 전달하고, IS로부터 여러 개의 $B_j,...,B_k$를 할당 받는다. 사용자는 *MsgGen* 알고리즘에서 여러 개의 $Y_2$ 메시지를 생성하고, SP는 *Authorization* 알고리즘의 (2) 과정을 반복하여 사용자의 다중 권한을 확인하는 것이 가능하다. 예를 들어, 1과 2의 권한을 가지는 사용자는 $B_1 = A^{\lambda_1}$과 $B_2 = A^{\lambda_2}$를 할당 받았다고 가정하자. 사용자는 *MsgGen* 알고리즘을 통해 $Y_1 = g_2^k, Y_2 = e(B_1^r, g_2^x w^r), Y_3 = e(B_2^r, g_2^x w^r)$을 계산하여 $Y=(Y_1, Y_2, Y_3)$ 메시지를 생성한다. 익명 인증을 위한 서명 값과 함께 SP에게 $Y$메시지가 전달되면, SP는 *Authorization* 알고리즘을 통해 $e(g_1^{\lambda_1}, Y_1) = Y_2$와 $e(g_1^{\lambda_2}, Y_1) = Y_3$을 검사하여 사용자의 다중 권한(권한 1과 권한 2)을 확인할 수 있다. 또한 사용자가 이전에 발급받은 권한 이외에 추가적으로 권한 발급이 필요한 경우에도 동일한 방법으로 권한 추가 발급이 가능하다.

만약 사용자가 이전에 소유한 권한 보다 더 넓은 범위의 권한으로 변경이 필요한 경우에는 그룹 서명 기법의 멤버쉽 비밀키 $(A,x,\tau)$는 전혀 수정할 필요 없이 사용자가 AM에게 새로운 권한 값을 할당받고 IS에게 새로운 권한 값을 위한 $B$를 받음으로써 해결 가능하다. 하지만, 더 좁은 범위의 권한 혹은 전혀 다른 권한으로의 변경이 필요한 경우에는 이전 권한에 대한 취소가 가능해야 한다. 즉, 사용자가 취소된 권한을 통해 인가를 받는 것을 막을 수 있어야 한다. 이를 위해, 사용자의 권한 취소 및 변경이 필요한 경우 멤버쉽 취소가 함께 이루어질 수 있도록 할 수 있다. 하지만 이 방법은 권한 취소 시마다 멤버쉽 취소 및 재발급이 이루어져야 하므로 불필요한 오버헤드가 많이 발생한다.

따라서 사용자의 멤버쉽은 유지되면서 권한 취소가 가능한 방법을 간단히 소개한다.

RA 혹은 다중 AM이 존재하는 경우에는 각각의 AM이 멤버쉽 RL과 별도로 권한에 대한 RL인 ARL (Authorization Revocation List)을 제공하여 멤버쉽 수정 없이 이전 권한만을 취소할 수 있다. 예를 들어, 사용자 $i$가 이전에 발급받은 권한 값이 $\lambda_7$이고, 이 권한을 취소하고 다른 권한을 위한 $\lambda_4$으로 변경해야 한다고 가정한다. $i$는 AM에게 이를 요청하고 4.3.2.1의 *A_Issue* 알고리즘을 수행한다. 이때 권한 연결 값 $\tau_i$는 수정 없이 이전의 것을 그대로 사용한다. 사용자는 발급받은 새로운 권한 값을 OP에게 등록하고, OP가 서명한 메시지($i$의 권한 변경 내용과 새로이 발급받은 권한 값을 포함한 메시지)를 받아 IS에게 전달한다. 이를 받은 IS는 *M_Issue* 알고리즘에서 (3), (4) 과정을 통해 새로운 $B_i' = A_i^{\lambda_4}$를 생성 및 발급한다. 사용자 $i$는 앞으로 새로이 발급받은 $ask'[i] = B_i'$를 이용하여 익명 인가를 받을 수 있다. RA 혹은 AM은 $i$가 $B_i = A_i^{\lambda_7}$를 통해 인가를 받는 것을 방지하기 위해, ARL을 사용한다. ARL은 특정 권한 발급 기관 별로 생성할 수 있으며, 다음과 같이 구성할 수 있다. 즉, AM의 *AUTH*의 권한 인덱스 별로 해당 권한의 취소 사용자에 대한 값 $g_2^{\lambda_j \tau_i}$을 포함한다.

$$ARL_{AM} = \left\{(1 : g_2^{\lambda_1 \tau_{i_1}}, g_2^{\lambda_1 \tau_{i_2}}, \ldots), \ldots, (j : g_2^{\lambda_j \tau_{i_1}}, g_2^{\lambda_1 \tau_{i_2}}, \ldots), \ldots\right\}$$

이전의 예에서는 $i$가 인덱스 7의 권한을 취소하므로 ARL의 인덱스 7에 대한 부분에 $g_2^{\lambda_7 \tau_i}$ 가 ARL에 추가되어 $ARL_{AM} = \left\{\ldots, (7 : \ldots, g_2^{\lambda_7 \tau_i}, \ldots), \ldots\right\}$이 된다. SP는 익명 사용자의 권한 검증 후, 권한이 취소된 자인지 아닌지를 ARL의 해당 권한에 포함된 값 $g_2^{\lambda_7 \tau_i}$과 AL의 해당 권한 검증을 위한 값 $g_1^{\lambda_7}$, 서명에 포함된 값들 $T_4, Y_1, Y_2$를 통해 $e(g_1^{\lambda_7}, Y_1)$와 $e(T_4, g_2^{\lambda_7 \tau_i})$가 같은지 검사하거나 $Y_2$와 $e(T_4, g_2^{\lambda_7 \tau_i})$가 같은지 검사하여 검증할 수 있다. ARL에 포함된 값을 통해서는 사용자의 비밀 권한 연결 값인 $\tau$를 알 수 없으므로 사용자의 익명성은 유지하면서 멤버쉽 비밀키 수정 혹은 멤버쉽 취소 없이 권한 취소 및 변경이 가능하다. 검증 과정은 다음과 같다.

$$e(g_1^{\lambda_7}, Y_1) = e(g_1^{\lambda_7}, g_2^k) = e(g_1, g_2)^{\lambda_7 k' \tau_i} = e(T_4, g_2^{\lambda_7 \tau_i})$$
$$Y_2 = e(B^k, g_2^{x_i} w^{\tau_i}) = e(g_1^{k\lambda_7}, g_2) = e(g_1, g_2)^{\lambda_7 k' \tau_i} = e(T_4, g_2^{\lambda_7 \tau_i})$$

## VI. 결론 및 향후과제

본 논문에서는 짧은 그룹 서명 기법을 활용하여 익명 인증뿐만 아니라 사용자의 권한에 따라 서비스 제공이 가능하도록 익명 인가가 가능한 기법을 제안하였다. 해당 기법은 권한 연결 값을 이용하여 권한과 멤버쉽을 연결하여 권한 위조를 위한 사용자 간의 공모가 불가능하도록 하였다. 사용자의 권한 수정 및 추가가 가능하며, 짧은 그룹 서명과 마찬가지로 특정 사용자의 익명성 취소가 다른 나머지 멤버들의 익명성에 영향을 미치지 않도록 나머지 멤버들의 그룹 공개키, 멤버쉽 비밀 키 수정이 가능하도록 하였다.

다양한 접근 제어 모델을 직접 적용하는 부분, 권한 취소 및 변경, 권한 위조 및 공모의 불가능성에 대한 자세한 증명을 향후 과제로 남겨두었다.

## 참고문헌


[1] G. Ateniese, J. Camenisch, M. Joye and G. Tsudik "A practical and provably secure coalition-resistant group signature scheme," In Proc. of *Crypto 2000*, LNCS 1880, Springer-Verlag, pp. 255-270, 2000.

[2] V. Benjumea, Choi Seung G., J. Lopez and M. Yung, "Anonymity 2.0 - X.509 Extensions Supporting Privacy-Friendly Authentication," In Proc. of *CANS 2007*, LNCS 4856, Springer-Verlag, pp. 265-281, 2007.

[3] D. Boneh and X. Boyen, "Short Signatures Without Random Oracles," In Proc. of *Eurocrypt 2004*, LNCS 3027, Springer-Verlag, pp. 56-73, 2004.

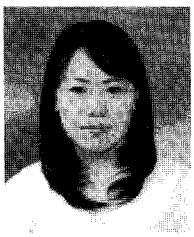
[4] D. Boneh, X. Boyen and H. Shacham, "Short group signatures," In Proc. of *Crypyo 2004*, LNCS 3152, Springer-Verlag, pp. 41-55, 2004.

[5] J. Camenisch, "Efficient and generalized group signatures," In Proc. of *Eurocrypt 1997*, LNCS 1233, Springer-Verlag, pp. 465-479, 1997.

[6] J. Camenisch and M. Stadler, "Efficient group signature schemes for large groups," In Proc. of *Crypto 1997*, LNCS, 1296,

Springer-Verlag, 1997.
[7] J. Camenisch and A. Lysyanskaya, "Signature schemes anonymous credentials from bilinear maps," In Proc. of *Crypto 2004*, LNCS 3152, Springer-Verlag, pp. 56-72, 2004.
[8] D. Chaum and E. van Heyst, "Group signatures," In Proc. of *Eurocrypt 1991*, LNCS 547, Springer-Verlag, pp. 257-265, 1991.
[9] L. Chen and T. P. Pederson, "New group signature schemes," In Proc. of *Eurocrypt 1994*, LNCS 950, Springer-Verlag, pp. 171-181, 1994.
[10] S. G. Choi, K. Park, and M. Yung, "Short traceable signatures based on bilinear pairings," In Proc. of *IWSEC 2006*, LNCS 4266, Springer-Verlag, pp. 88-103, 2006.
[11] A. Kiayias, Y. Tsiounis, and M. Yung, "Traceable Signatures," In Proc. of *Eurocrypt 2004*, LNCS 3027, Springe-Verlag, pp. 571-589, 2004.
[12] A. Lysyanskaya, R. Rivest, A. Sahai, and S. Wolf, "Pseudonym systems," In Proc. of *SAC 1999*, LNCS 1758, Springer-Verlag, pp. 184-199, 1999.
[13] L. Nguyen and R. Safavi-Naini, "Efficient and provably secure trapdoor-free group signature schemes from bilinear pairings," In Proc. of *Asiacrypt 2004*, LNCS 3329, Springer-Verlag, pp. 372-386, 2004.
[14] D. Yao and R. Tamassia, "Compact and Anonymous Role-Based Authorization Chain," ACM Trans. on Information and System Security (TISSEC), Vol. 12, No. 3, 2009.

## 〈著者紹介〉



신 수 연 (Sooyeon Shin) 학생회원
2004년 2월: 세종대학교 컴퓨터공학과 학사
2006년 2월: 세종대학교 컴퓨터공학과 석사
2006년 9월 ~ 현재: 세종대학교 컴퓨터공학과 박사과정
〈관심분야〉 프라이버시 보호기술, 익명성 기술, RFID, 센서 네트워크 보안, HCI 보안 등

권 태 경 (Taekyoung Kwon) 종신회원
1992년 2월: 연세대학교 컴퓨터과학과 학사
1995년 2월: 연세대학교 컴퓨터과학과 석사
1999년 8월: 연세대학교 컴퓨터과학과 박사
1999년 ~ 2000년: U.C. Berkely Post-Doc.
2001년 ~ 현재: 세종대학교 컴퓨터공학과 부교수, 정보보호학회 이사 및 편집위원
〈관심분야〉 암호프로토콜, 네트워크 프로토콜, 센서네트워크 보안, 프라이버시 보호, HCI 보안 등