A Study on Privacy-Enhancing User Authentication Systems with Efficiency Improvements

by

SHAHIDATUL SADIAH BINTI ABDUL MANAN
D152447

A dissertation submitted in partial fulfilment of
the requirements for the degree of
Doctor of Engineering

Under Supervision of
Professor Toru Nakanishi

Department of Information Engineering
Graduate School of Engineering
Hiroshima University

March, 2018

Abstract

The internet has currently become a ubiquitous channel for data hub and dissemination since the networking infrastructure has grown to connect computers all over the world through cloud applications and online services. In such services, user authentications are required to permit only access from a valid user. On the other hand, through the authentication, service providers collect a large amount of information about the users and their online activities. This information can be beneficial for the service providers, but the way of handling them might also present challenges to user's privacy.

One of cryptographic solutions to protect the users' privacy in the authentication is an *anonymous credential scheme*. This scheme allows an issuer to issue each user a certificate as a proof of the qualification that contains the user's attributes. The user can anonymously convince any verifier for the possession of the certificate, where the selected attributes can be disclosed without revealing any other information about the user's privacy.

In general, complex relations on attributes can be proved. Previously, an anonymous credential system with constant size proofs was proposed, where a user can prove any Conjunctive Normal Form (CNF) formulas, i.e., an ANDs of ORs, on attributes. However, this system still suffers from inefficiency in the case of numerous OR literals, due to the less expression capability of CNF formulas. To achieve the constant-size proof, this system utilizes an *accumulator* that compresses multiple attributes of a formula into a single value. In the compression, the accumulator requires that all public parameters assigned to the attribute values of OR literals in the formula are multiplied which cause a large delay in the proof generation.

In the first part of this thesis, we propose an extended accumulator to prove *monotone formulas* on attributes and apply it to the anonymous credential system in order to obtain more efficiency in the proof generation. The monotone formula is a logic formula that contains any combination of AND and OR relations without negations. That is, CNF formula is a limited type of the monotone formulas. Thus, in the cases that proved formulas require longer sizes in the representation of CNF formula than monotone formula, the proposed system has more efficient computation costs. We ensure this in the implementation, where the experimental result shows that the proposed scheme reduced the proving time from 969.11ms to 63.04ms and the verifying time is reduced from 376.97ms to 132.57ms in a practical example.

Another cryptographic solution to protect the users' privacy in the authentication is a *group signature scheme*, which is the digital signature version of the anonymous credential. In the scheme, a group member is allowed to sign a message anonymously on behalf of the group. There are two types of authorities engage: A *group manager*, (GM) who adds users into the group, and an *opener* who can identify the signer from the signature when necessary. One important function in the group signature scheme is *revocation*, where the user's privilege to sign a message is removed. The revocation is a critical issue, which has been broadly studied.

Previously, Libert et al. proposed a revocable group signature scheme, where for number of users N, the scheme has achieved $O(1)$ signature size, $O(1)$ signing/verification costs, $O(1)$ membership certificate size, and $O(\log N)$ public key size. However, the scheme still needs an improvement on the $O(R)$ revocation list (RL) size, where R is the number of revocations. This is because the signer needs to fetch the RL for every revocation epoch, thus, the large size will cause delay in mobile environment. Later, Nakanishi et al. proposed a scheme with compact RL using an accumulator. In this scheme, GM accumulates T subsets in the SD method and signs the accumulated value for any integer T. Thus, the number of signatures is reduced by $1/T$ and the RL size is $O(R/T)$. However, the signing time, the public key size and membership certificate size are increased, when T is increased.

On the second part of this paper, we extend the scheme proposed by Libert et al.. In the proposed scheme, similarly to the scheme by Nakanishi et al., we partition the subsets into a number of blocks and compress it using a *vector commitment*. Since the compression is simpler than the accumulator, we can reduce the RL size to $O(R/T)$ while maintaining the membership certificate size as $O(1)$. However, the signing cost still depends on T, and the verification has constant overhead costs, since there are more proofs of the zero-knowledge fashion. This scheme seems to be practical on the RL size, but the practicality of the signing time for concrete values of T, and the overheads in the verification time are unknown. To clarify that, we implemented the scheme with some efficiency improvements to show the time efficiency. From the experimental results, the signing time is less than 500 ms for $T = 400$, but the verification time is about 1.5 s. We consider that the implemented scheme is practical in a mobile environment due to lower user computation time and storage.

List of Publications

Referred Journals

- S. Sadiah and T. Nakanishi,“Revocable Group Signatures with Compact Revocation List Using Vector Commitments,” IEICE Trans Fundamentals, Vol.E100-A, No.8, pp. 1672-1682, August 2017.
- S. Sadiah, T. Nakanishi, N. Begum, and N. Funabiki,“Accumulator for Monotone Formulas and its Application to Anonymous Credential System,” Journal of Information Processing, Vol.25, pp. 949-961, December 2017.

Referred International Conferences

- S. Sadiah, T. Nakanishi, and N. Funabiki, “Anonymous Credential System with Efficient Proofs for Monotone Formulas on Attributes,” IWSEC 2015, LNCS 9241, pp. 262-278, August 2015.
- S. Sadiah and T. Nakanishi, “Revocable Group Signatures with Compact Revocation List Using Vector Commitments,” WISA 2016, LNCS 10144, pp. 245-257, August 2016.
- S. Sadiah and T.Nakanishi, “Implementation of Revocable Group Signatures with Compact Revocation List Using Vector Commitments”, CANDAR 2017, pp. 489-495.

Other International Conferences

- S. Sadiah, T. Nakanishi, and N. Funabiki, “Implementation of Anonymous Credential System with Efficient Proofs for Monotone Formulas on Attributes Excluding Restriction”, CANDAR 2014.
- S. Sadiah and T. Nakanishi, “Reduction of Certificates in an Anonymous Credential System with Proofs for Monotone Formulas on Attributes”, ICCE-TW 2016.

Contents

Chapter 1

Introduction

1.1 Backgrounds

In current digital era, the internet has now become an ubiquitous channel for data hub and dissemination. The networking infrastructure has grown to connect computers all over the world through cloud applications and online services. In such services, user authentications are required to permit only access from a valid user. However, through the authentication, service providers collect a large amount of informations about the users and their online activities. These informations can be beneficial for the improvements of the services, but the way of handling them might also present challenges to user's privacy.

One of cryptographic solutions to protect the users' privacy in the authentication is an *anonymous credential scheme*. There exist three entities in the anonymous credential scheme: an issuer, users, and the verifier (i.e., service provider). This scheme allows an issuer to issue a certificate to a user. Each certificate is a proof of the membership, the qualification, or the privilege, and contains user's attributes. The user can anonymously convince any verifier for the possession of the certificate, where the selected attributes can be disclosed without revealing any other information about the user's privacy.

In general, complex relations on attributes can be expressed by logic formulas. The AND relation is used when proving the possession of all the multiple attributes. The OR relation represents the possession of one of the multiple attributes. For example, when accessing an alcohol-related company's website, most companies would ask for both nationality and birthday attributes (to prove the age) during the authentication, since different countries have a different legal drinking age. Thus, the user needs to prove the AND relation of his/her nationality and age. In the authentication, a zero-knowledge type of proof allows the user to hide any other information beside the satisfaction of the relations.

In [23], an anonymous credential system with constant size proofs was proposed, where a user can prove any Conjunctive Normal Form (CNF) formulas, i.e., an ANDs of ORs, on attributes. However, this system still suffers from inefficiency in the case of numerous OR literals, due to the less expression capability of CNF formulas. To achieve the constant-size proof, this system utilizes an *accumulator* that compresses multiple attributes of monotone formula into a single value. In the compression, the accumulator requires that all public parameters assigned to the attribute values of OR literals in the formula are multiplied which cause a large delay in the proof generation.

Another cryptographic solution to protect the users' privacy in the authentication is a *group signature scheme*, which is a signature scheme version of the anonymous credential scheme. The authentication is an online procedure between a user and a verifier, while the signature can be verified off-line by one or probably more verifiers. In the group signature scheme, the signer (i.e., user) signs a message on behalf of the group, and the verifier (i.e., service provider) verifies the signature anonymously. There are two types of authorities: A *group manager* (GM) who adds users into the group, and an *opener* who can identify the signer from the signature when necessary. In the user registration phase, GM issues a membership certificate to the user. Then, the user creates the group signature using the certificate. Any verifier can check the validity of the signature by using the group public key without knowing who is the actual user. As an additional functionality in the group signature scheme, *revocation* has been introduced, where the user's privilege to sign a message is removed. It is a critical issue due to the anonymity of the signature and has been broadly studied.

Previously, a scalable group signature scheme with revocation based on the broadcast encryption framework was proposed by Libert et al. [4]. Despite achieving efficient complexity costs of $O(1)$ signature size, $O(1)$ signing/verification costs, $O(1)$ membership certificate size, and $O(\log N)$ public key size, where N is the total number of group members, the scheme still needs an improvement on the revocation list (RL) size. In the scheme, the RL contains signatures for all subsets of authorized users, which are formed by a subset difference (SD) method. In the worst case, the number of signatures amounts to $2R-1$, where R is the number of revocations. The used signature scheme is an AHO signature with 7 group elements. Thus, in case of 128-bit security, the RL size is $900R$ bytes or more. The large size will cause delay in a mobile environment, since the signer needs to fetch the RL for every revocation epoch.

Both of the above schemes are legitimate solutions to the user's privacy problem. However, as mentioned before, the efficiency of both for practical use are still questionable. In this thesis, we address problems from two areas: (i) computational efficiency of the attribute-based anonymous credential scheme, and (ii) data size of the revocable group signature scheme.

1.2 Our Contributions

On the first part, we deal with the time efficiency in the proof generation of previous work [23]. We propose an extended accumulator to prove *monotone formulas* on attributes and apply it to the anonymous credential system in order to obtain more efficiency in the proof generation. The monotone formula is a logic formula that contains any combination of AND and OR relations without negations. That is, the CNF formula is a limited type of the monotone formulas.

The previous work suffers from inefficiency in the case of numerous OR literals, due to the less expression capability of CNF formulas. A typical example is to prove the age using birthday attributes. Consider the example in accessing an alcohol related website for countries that have a legal drinking age of 18 years old or above. An example of CNF formula is $(Australia \vee \ldots) \wedge (1915, Jan.1^{st} \vee \cdots \vee 1997, Sept.5^{th})$, where each birthday is encoded to one attribute value such as "$1915, Jan.1^{st}$". The accumulator requires that all public parameters assigned to the attribute values of OR literals in the formula are multiplied. For the above example, there are 101 attributes for nationality, and the number of attributes for the birthdays from $1915, Jan.1^{st} \sim 1997, Sept.5^{th}$ is $30,198$, which makes $30,299$ attributes in total. The multiplications cause a large delay in the authentication.

The expression flexibility of monotone formula can reduce the number of attributes in the proved formula as follows. In the monotone formula, the birthday attribute is composed of the birth-year, the birth-month, and the birth-day, and one birthday is expressed as (birth-year $\wedge$ birth-month $\wedge$ birth-day). For the above example, the monotone formula is $(Australia \vee \ldots) \wedge (1915 \vee \ldots \vee (1997 \wedge (Jan. \vee \ldots \vee (Sept. \wedge (1^{st} \vee \ldots \vee 5^{th})))))$. Using this type of formula, the number of public parameters multiplied in the accumulator is decreased to 198 attributes in total of 101 nationalities, 83 birth-years, 9 birth-months, and 5 birth-days, which greatly impacts the reduction of authentication time.

On the second part, we deal with the data size of revocation list (RL) of previous scheme [4]. In the scheme, the RL contains signatures for all subsets of authorized users, which are formed by a subset difference (SD) method. In the worst case, the number of signatures amounts to $2R - 1$, where R is the number of revocations. As the signature, an AHO signature with 7 group elements is used. Thus, in case of 128-bit security, the RL size is $900R$ bytes or more. Since the signer needs to fetch the RL for every revocation epoch, the large size will cause delay in mobile environment.

Thus, we propose a revocable group signature scheme with a compact RL to overcome the problem. by partitioning the subsets into a number of blocks and compress it using a vector commitment [2]. We can reduce the RL size to $O(R/T)$, where T is the number of compressions, and the public key size is $O(T + \log N)$, where N is the total number of group members, while maintaining the membership certificate size as $O(1)$. In the proposed scheme, for $R = 100,000$, the size of RL is reduced from 63KB in [4] to 1,400KB.

However, as the trade-off, the signing cost becomes $O(T)$. In this thesis, we also show the experimental result of the proposed scheme based on the implementation to clarify the practicality.

1.3 Contents of This Thesis

The remaining of this thesis is organized as follows.

Chapter 2 reviews the mathematics fundamentals for this thesis. This chapter covers the introduction of the bilinear maps, and the basic concept of pairings. Then, the complexity assumptions and signature scheme are reviewed, and zero-knowledge proof technique used in this thesis are illustrated in two types, the symmetric and the asymmetric types.

Chapter 3 proposes the construction of a anonymous credential scheme for monotone formula on attributes. Before the construction, as the key component, extended accumulator to verify the monotone formula in the anonymous authentication is proposed. This chapter also illustrates the comparisons of the signing time and the verification time between the proposed scheme and the the previous work.

Chapter 4 proposes the construction of a revocable group signature scheme with compact revocation list using a vector commitment. This chapter also shows the theoretical efficiency comparisons with the previous works.

Chapter 5 describes the implementation of the proposed group signature scheme to show the efficiency, which includes efficiency improvements for the implementation based on pairing. This chapter also discusses the experimental results for the consideration of practicality.

Finally, Chapter 6 concludes this thesis together with some future works.

Chapter 2

Preliminaries

In this chapter, we review the bilinear map, the complexity assumptions, and the cryptograhic tools used as building blocks of our proposed systems.

2.1 Bilinear maps

The concepts related to bilinear maps in our system, which are implemented by the pairing, are as follow:

1. $\mathbb{G}_1$ and $\mathbb{G}_2$ are two multiplicative cyclic groups of prime order p;
2. g_1 and g_2 are randomly chosen generator of $\mathbb{G}_1$, $\mathbb{G}_1$, respectively;
3. e is a computable bilinear map, $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with the following properties:
    - Bilinearity: for all $u, v \in \mathbb{G}_1$, and $a, b \in \mathbb{Z}, e(u^a, v^b) = e(u, v)^{ab}$.
    - Non-degeneracy: $e(g_1, g_2) \neq 1_{\mathbb{G}_T}$ where $1_{\mathbb{G}_T}$ is an identity element of $\mathbb{G}_T$.

The above map is implemented by the asymmetric pairing. However, in Chapter 4, for simplicity, we propose the revocable group signature scheme using the symmetric pairing as the bilinear map, where e is defined as $\mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$.

2.2 Complexity assumptions

On the first part of the thesis, the security of our proposed anonymous credential scheme is based on SXDH assumption, the n-DHE (DH Exponent) assumption [14] and the q-SFP (Simultaneous Flexible Pairing) assumption [19]. The assumptions are define in asymmetric type pairing.

Definition 1. (SXDH assumption) *The decisional Diffie-Hellman assumption holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$.*

Definition 2. (n-DHE assumption) *For all PPT algorithm $\mathcal{A}$, the probability*

$$Pr\left[\mathcal{A}\left(\ g, g^a, \ldots, g^{a^n}, g^{a^{n+2}}, \ldots, g^{a^{2n}}, \tilde{g}, \tilde{g}^a, \ldots, \tilde{g}^{a^n}, \tilde{g}^{a^{n+2}}, \ldots, \tilde{g}^{a^{2n}}\ \right) = \tilde{g}^{a^{n+1}}\right]$$

is negligible, where $g \in_R \mathbb{G}_1$, $\tilde{g} \in_R \mathbb{G}_2$ and $a \in_R \mathbb{Z}_p$.

Definition 3. **(q-SFP assumption)** *For all PPT algorithm $\mathcal{A}$, the probability*

$$Pr\left[\begin{array}{l} \mathcal{A}\big(\ g_z, h_z, g_r, h_r, a, \tilde{a}, b, \tilde{b}, \{(z_j, r_j, s_j, t_j, u_j, v_j, w_j)\}_{j=1}^{q}\ \big) \\ = (z^*, r^*, s^*, t^*, u^*, v^*, w^*) \wedge\ e(a,\tilde{a}) = e(g_z, z^*)e(g_r, r^*)e(s^*, t^*)\ \wedge \\ e(b,\tilde{b}) = e(h_z, z^*)e(h_r, u^*)e(v^*, w^*) \wedge\ z^* \neq 1\ \wedge\ z^* \neq z_j \textit{ for all } 1 \leq j \leq q \end{array}\right]$$

is negligible, where $(g_z, h_z, g_r, h_r) \in \mathbb{G}_1^4$, $(a, \tilde{a})$ and $(b, \tilde{b})$ be pairs in $\mathbb{G}_1 \times \mathbb{G}_2$, and all tuples $\{s_j, v_j\}_{j=1}^{q} \in \mathbb{G}_1^2$ and $\{z_j, r_j, u_j, t_j, w_j\}_{j=1}^{q} \in \mathbb{G}_2^5$ satisfy the above relations.

On the second part of the thesis, the security of our proposed revocable group signature scheme is based on DLIN (Decision LINear) assumption [6], the q-SDH (Strong DH) assumption [7], the n-FlexDHE assumption [21], the n-DHE assumption, and the q-SFP (Simultaneous Flexible Pairing) assumption [19]. The assumptions are define in symmetric type pairing.

Definition 1 (DLIN assumption) For all PPT algorithm $\mathcal{A}$, the probability

$$\left|\ Pr\left[\mathcal{A}(g, g^a, g^b, g^{ac}, g^{bd}, g^{c+d}) = 1\right] - Pr\left[\mathcal{A}(g, g^a, g^b, g^{ac}, g^{bd}, g^{z}) = 1\right]\ \right|$$

is negligible, where $g \in_R \mathbb{G}$ and $a, b, c, d, z, \in_R \mathbb{Z}_p$.

Definition 2 (q-SDH assumption) For all PPT algorithm $\mathcal{A}$, the probability

$$Pr\left[\mathcal{A}(g, g^a, \ldots, g^{a^q}) = (b, g^{1/(a+b)} \wedge b \in \mathbb{Z}_p)\right]$$

is negligible, where $g \in_R \mathbb{G}$ and $a \in_R \mathbb{Z}_p$.

Definition 3 (n-DHE assumption) For all PPT algorithm $\mathcal{A}$, the probability

$$Pr\left[\mathcal{A}(g, g^a, \ldots, g^{a^n}, g^{a^{n+2}}, \ldots, g^{a^{2n}}) = g^{a^{n+1}}\right]$$

is negligible, where $g \in_R \mathbb{G}$ and $a \in_R \mathbb{Z}_p$.

Definition 4 (n-FlexDHE assumption) For all
PPT algorithm $\mathcal{A}$, the probability

$$Pr\left[\ \mathcal{A}(g, g^{a^1}, \ldots, g^{a^n}, g^{a^{n+2}}, \ldots, g^{a^{2n}}) = (g^{\mu}, g^{\mu a^{n+1}}, g^{\mu a^{2n}}) \wedge \mu \in \mathbb{Z}_p\ \right]$$

is negligible, where $g \in_R \mathbb{G}$ and $a \in_R \mathbb{Z}_p$. The n-FlexDHE assumption is stronger than the n-DHE assumption, i.e., the former implies the latter.

Definition 5 (q-SFP assumption) For all PPT algorithm $\mathcal{A}$, the probability

$$Pr\left[\begin{array}{l} \mathcal{A}\big(\ g_z, h_z, g_r, h_r, a, \tilde{a}, b, \tilde{b}, \{(z_j, r_j, s_j, t_j, u_j, v_j, w_j)\}_{j=1}^{q}\ \big) \\ = (z^*, r^*, s^*, t^*, u^*, v^*, w^*) \in \mathbb{G}^7 \wedge\ e(a,\tilde{a}) = e(g_z, z^*)e(g_r, r^*)e(s^*, t^*)\ \wedge \\ e(b,\tilde{b}) = e(h_z, z^*)e(h_r, u^*)e(v^*, w^*) \wedge\ z^* \neq 1\ \wedge\ z^* \neq z_j \text{ for all } 1 \leq j \leq q \end{array}\right]$$

is negligible, where $(g_z, h_z, g_r, h_r, a, \tilde{a}, b, \tilde{b}) \in \mathbb{G}^8$ and all tuples $\{(z_j, r_j, s_j, t_j, u_j, v_j, w_j)\}_{j=1}^{q}$ satisfy the above relations.

2.3 AHO Structure-Preserving Signatures

AHO signature [19] is used as the structure-preserving signature, where the knowledge of the signature can be proved by the following Groth-Sahai (GS) proofs. The AHO signature allows us to sign multiple elements to obtain a signature with the constant size.

AHOKeyGen: Select bilinear groups $\mathbb{G}, \mathbb{G}_T$ with a prime order p and a bilinear map e. Select $g, G_r, H_r \in \mathbb{G}$ and $\mu_z, \nu_z, \mu_1, \ldots, \mu_k, \nu_1, \ldots, \nu_k, \alpha_a, \alpha_b \in_R \mathbb{Z}_p$. Compute $G_z = G_r^{\mu_z}, H_z = H_r^{\nu_z}, G_1 = G_r^{\mu_1}, \ldots, G_k = G_r^{\mu_k}, H_1 = H_r^{\nu_1}, \ldots, H_k = H_r^{\nu_k}, A = e(G_r, g^{\alpha_a})$, $B = e(H_r, g^{\alpha_b})$. Output the public key as $pk = (\mathbb{G}, \mathbb{G}_T, p, e, g, G_r, H_r, G_z, H_z, G_1, \ldots, G_k, H_1, \ldots, H_k, A, B)$, and the secret key as $sk = (\alpha_a, \alpha_b, \mu_z, \nu_z, \mu_1, \ldots, \mu_k, \nu_1, \ldots, \nu_k)$.

AHOSign: Given a vector of messages $(M_1, \ldots, M_k) \in \mathbb{G}^k$ together with sk, choose β, ϵ, η, ι, $\kappa \in_R \mathbb{Z}_p$, and compute $\theta_1 = g^\beta$, and

$$\theta_2 = g^{\epsilon - \mu_z \beta} \prod_{i=1}^k M_i^{-\mu_i},\ \theta_3 = G_r^\eta,\ \theta_4 = g^{(\alpha_a - \epsilon)/\eta},$$
$$\theta_5 = g^{\iota - \nu_z \beta} \prod_{i=1}^k M_i^{-\nu_i},\ \theta_6 = H_r^\kappa,\ \theta_7 = g^{(\alpha_b - \iota)/\kappa}.$$

Output the signature $\sigma = (\theta_1, \ldots, \theta_7)$.

AHOVerify: Given a vector of messages $(M_1, \ldots, M_k) \in \mathbb{G}^k$ and the signature $\sigma = (\theta_1, \ldots, \theta_7)$, accept these if following equations are hold:

$$A = e(G_z, \theta_1) \cdot e(G_r, \theta_2) \cdot e(\theta_3, \theta_4) \cdot \prod_{i=1}^k e(G_i, M_i),$$
$$B = e(H_z, \theta_1) \cdot e(H_r, \theta_5) \cdot e(\theta_6, \theta_7) \cdot \prod_{i=1}^k e(H_i, M_i).$$

This signature is existentially unforgeable against chosen-message attacks under the q-SFP assumption [19].

The re-randomization algorithm in [19] allows us to publicly randomize an AHO signature to obtain another signature $(\theta'_1, \ldots, \theta'_7)$ on the vector of the same messages. In the GS proof of the randomized signature, $(\theta'_i)_{i=3,4,6,7}$ can be revealed, but $(\theta'_i)_{i=1,2,5}$ is committed.

2.4 Groth-Sahai (GS) Proof

To prove the secret knowledge in relations of the bilinear maps, we utilize Groth-Sahai (GS) proofs [16]. We adopt the instantiation based on SXDH assumption. For the bilinear groups, the proof system needs a common reference string $(\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{v}_1, \boldsymbol{v}_2)$ for $\boldsymbol{u}_1 = (u_{11}, u_{12})$, $\boldsymbol{u}_2 = (u_{21}, u_{22})$, $\boldsymbol{v}_1 = (v_{11}, v_{12})$, $\boldsymbol{v}_2 = (v_{21}, v_{22})$ for some $u_{11}, u_{12}, u_{21}, u_{22} \in \mathbb{G}_1$ and some $v_{11}, v_{12}, v_{21}, v_{22} \in \mathbb{G}_2$. The commitment to an element $X \in \mathbb{G}_1$ (resp., $Y \in \mathbb{G}_2$) is computed as $\boldsymbol{C} = (1, X) \cdot \boldsymbol{u}_1^r \cdot \boldsymbol{u}_2^s$ (resp, $\boldsymbol{C} = (1, Y) \cdot \boldsymbol{v}_1^r \cdot \boldsymbol{v}_2^s$) for $r, s \in_R \mathbb{Z}_p^*$. In the case of the CRS setting for perfectly sound proofs, $\boldsymbol{u}_2 = \boldsymbol{u}_1^{\xi_1}$, $\boldsymbol{v}_2 = \boldsymbol{v}_1^{\xi_2}$ for $\xi_1, \xi_2 \in_R \mathbb{Z}_p^*$. Then, the commitment $\boldsymbol{C} = (u_{11}^{r+\xi_1 s}, X \cdot u_{12}^{r+\xi_1 s})$ (resp., $\boldsymbol{C} = (v_{11}^{r+\xi_2 s}, Y \cdot v_{12}^{r+\xi_2 s})$) is the ElGamal encryption for X (resp., Y). On the other hand, in the setting of the witness indistinguishability, $\boldsymbol{u}_2 = \boldsymbol{u}_1^{\xi_1}/(1, g)$, $\boldsymbol{v}_2 = \boldsymbol{v}_1^{\xi_2}/(1, \tilde{g})$ for $\xi_1, \xi_2 \in_R \mathbb{Z}_p^*$, and thus $\boldsymbol{C}$ is perfectly hidden. The SXDH assumption implies the indistinguishability of the CRS. To prove that the committed variables in the pairing relations, the prover prepares the commitments, and replaces the variables in the pairing relations by the commitments. The GS proof allows us to prove the set of pairing product equations:

$$\prod_{j=1}^{n} e(A_j, Y_j) \cdot \prod_{i=1}^{m} e(X_i, B_i) \cdot \prod_{i=1}^{m}\prod_{j=1}^{n} e(X_i, Y_j)^{a_{ij}} = t$$

for variables $X_1, \ldots, X_m \in \mathbb{G}_1, Y_1, \ldots, Y_n \in \mathbb{G}_2$ and constants $A_1, \ldots, A_n \in \mathbb{G}_1, B_1, \ldots, B_m \in \mathbb{G}_2, a_{ij} \in \mathbb{Z}_p, t \in \mathbb{G}_T$.

The GS proof system consists of the following algorithms: **SoundSetup** outputs a CRS crs for perfectly sound proofs together with the extraction trapdoor et. **ProofGen**, on input of crs, a statement of pairing relations S, and a witness W that is a set of committed variables satisfying the pairing relations, outputs the proof π including the commitments of W. **Verify**, on input of crs, and π, outputs the acceptance if the proof is valid, or rejection otherwise.

Furthermore, there are special algorithms: **Extract**, on inputs of crs, et, and π, outputs the witness W. **WISetup** outputs a CRS for the witness indistinguishability, crs'. **WIProofGen** on input of crs', a statement of pairing relations S, and a witness W for S, outputs the witness indistinguishable proof π'.

The GS proof system satisfies the following security properties.

CRS indistinguishability: crs output by **SoundSetup** and crs' output by **WISetup** are computationally indistinguishable.

Extractability (Perfect soundness): For crs and et output by **SoundSetup** and a proof π, if **Verify** outputs the acceptance on π, **Extract** can output the witness W from π.

Perfect witness indistinguishability (WI): Consider the following game:

Game$_{\mathbf{WI}}$:

1. The challenger runs **WISetup** to generate crs'. It gives crs' to the adversary.
2. The adversary outputs a statement S and the witnesses W_1, W_2. The challenger randomly selects $b \in_R \{0, 1\}$, and responds the proof using W_b on crs', S using **WIProofGen** to the adversary.
3. The adversary outputs the guess b'.

Then, for any PPT adversary in **Game$_{\mathbf{WI}}$**, $\Pr[b' = b] = 1/2$.

2.5 Tag-based Encryption

As in [4], we adopt the public key tag-based encryption [11]. The public key consists of random non-trivial elements $pk = (f_1, f_2, U, V) \in \mathbb{G}^4$ and the secret key is $sk = (\omega, \eta)$ where $f_1 = g^{\omega}, f_2 = g^{\eta}$. We encrypt message $M \in \mathbb{G}$ using tag $t \in \mathbb{Z}_p$ and randomness $r, s \in \mathbb{Z}_p$ as $(\Upsilon_1, \ldots, \Upsilon_5)$:= $(f_1^r, f_2^s, g^{r+s}M, (g^tU)^r, (g^tV)^s)$. The validity of the ciphertext is publicly verifiable, since valid ciphertexts have $e(f_1, \Upsilon_4) = e(\Upsilon_1, g^tU)$ and $e(f_2, \Upsilon_5) = e(\Upsilon_2, g^tV)$. Decryption can be done by computing $M = \Upsilon_3\Upsilon_1^{-1/\omega}\Upsilon_2^{-1/\eta}$. In the group signature scheme, we will set up the cryptosystem with the same f_1, f_2 as in the common reference string of the non-interactive proofs.

Under the DLIN assumption this cryptosystem is selective-tag weakly CCA-secure.

2.6 Vector Commitment

We adopt a primitive called vector commitment [2], where a vector of multiple values are committed, and the commitment can be opened at specific coordinate. Generally, the commitment is randomized for hiding. However, as in [4], we utilize the non-randomized version, since we need only the binding property. The public key for the commitments is $pk_{vc} = (g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n})$, where $g_i = g^{\gamma^i}$ for each $\gamma \in_R \mathbb{Z}_p$. To commit to a vector $\overrightarrow{m} = (m_1, \ldots, m_n) \in \mathbb{Z}_p^n$, the committer computes

$$C = \prod_{\kappa=1}^{n} g_{n+1-\kappa}^{m_\kappa}.$$

A single group element $W_i = \prod_{\kappa=1, \kappa \neq i}^{n} g_{n+1-\kappa+i}^{m_\kappa}$ provides the evidence that m_i is the i-th component of $\overrightarrow{m}$. It satisfies the verification relation $e(g_i, C) = e(g_1, g_n)^{m_i} \cdot e(g, W_i)$. The infeasibility of opening a commitment to two distinct messages for some coordinate i relies on the n-FlexDHE assumption or n-DHE assumption (see the *proof* of **Theorem 6**).

2.7 The NNL Framework for Broadcast Encryption

The Subset-Cover framework for broadcast encryption with $N = 2^\ell$ registered receiver (user) was proposed in [9]. In a complete binary tree $\mathtt{T}$ of height ℓ, each node is assigned a secret key, where the receivers are associated with the leaves. The framework's idea is to partition the set of non-revoked users into m disjoint subsets $S_1, \ldots, S_m$ such that $\mathcal{N} \setminus \mathcal{R} = S_1 \cup \ldots \cup S_m$, where $\mathcal{N}$ is the universe set of users and $\mathcal{R} \subset \mathcal{N}$ is the set of revoked receivers.

The Subset Difference (SD) method has $O(|\mathcal{R}|)$ transmission cost and $O(\log^2 N)$ storage complexity. For each node x in $\mathtt{T}$, the subtree rooted at x is called $\mathtt{T}_x$. For each $i \in 1, \ldots, m$, the disjoint subset S_x of $S_1, \ldots, S_m$ is defined by a node primary node, e.g., the root node of the subtree and a secondary node, e.g., a descendant of the primary node which its descendants are all revoked. Each user belongs to many generic subsets, so that the number of subsets bounded by $m = 2 \cdot |\mathcal{R}| - 1$, as proved in [9].

Chapter 3

Accumulator for Monotone Formulas and its Application to Anonymous Credential System

3.1 Introduction

In Web services, user authentications are required to protect malicious access. In conventional ID-based authentications, the privacy problem may occur, since Service Provider (SP) can trace the user's ID, grasp user's service history, and might use it to attempt malicious activities. On the other hand, from SP's point of view, the authentication using the user's attributes such as a gender, an occupation, and an age is more advantageous for commercial values. Thus, an attribute-based authentication with a strong privacy protection is in demand, where users can selectively disclose the minimal amount of attributes necessary for the service while hiding the others completely.

For the demand, in [12, 15, 1, 21], *anonymous credential systems* were proposed, where a user can anonymously convince SP about the possession of specified attributes. There exist three entities in the anonymous credential system: an issuer, users, and SP. The user obtains a certificate from the issuer in advance, where the certificate ensures his/her attributes. Then, the user makes a proof of the certified attributes and prove it to SP. In the authentication, SP requests the user to prove his/her certified attributes and their relation. For example, when accessing an alcohol-related company's website, most companies would ask for both nationality and birthday attributes (to prove the age) during the authentication, since different countries have different legal drinking age. Thus, the user needs to prove the AND relation of his/her nationality and age. In general, complex relations on attributes can be expressed by logic formulas. The AND relation is used when proving the possession of all the multiple attributes. The OR relation represents the possession of one of the multiple attributes. In the authentication, a zero-knowledge type of proof allows the user to hide any other information beside the satisfaction of the relations.

In [15, 1], anonymous credential systems were proposed, where the proof of the formula has the constant size for the number of all attributes of a user and the size of the proved formula. However, simple AND or OR relations on attributes are only available. In [21], a system with the constant size proofs is proposed, where the inner-product on attributes can be proved (Thus, CNF and DNF formulas are also available). However, in this system, the proof generation needs exponentiation depending on the number of literals in the OR relations, which causes a large delay in case of formulas with lots of OR literals.

In [23], an anonymous credential system with the constant size proofs was proposed, where a user can prove any CNF formulas on attributes. In this system, the proof generation is more efficient than [21], since it needs only multiplications depending the number of OR literals. However, this system still suffers from the inefficiency in case of numerous OR literals, due to the less expression capability of CNF formulas. The typical example is to prove the age using birthday attributes. To achieve the constant-size proof, this system utilizes an *accumulator* that compresses multiple attributes of monotone formula into a single value. In the compression, multiplications are needed. Consider the above example in accessing an alcohol related website for countries that have legal drinking age of 18 years old or above. An example of CNF formula is $(Australia \vee \ldots) \wedge (1915, Jan.1^{st} \vee \cdots \vee 1997, Sept.5^{th})$, where each birthday is encoded to one attribute value such as "$1915, Jan.1^{st}$". The accumulator requires that all public parameters assigned to the attribute values of OR literals in the formula are multiplied. For the above example, there are 101 attributes for nationality, and the number of attributes for the birthdays from $1915, Jan.1^{st} \sim 1997, Sept.5^{th}$ is $30,198$, which makes $30,299$ attributes in total. The multiplications cause a large delay in the authentication.

In this chapter, we propose an extended accumulator to prove *monotone formulas* on attributes and apply it to the anonymous credential system in order to obtain more efficiency in the proof generation than the previous system [23] for proving CNF formulas. The monotone formula is a logic formula that contains any combination of AND and OR relations without negations. That is, the CNF formula is a limited type of the monotone formulas. In the monotone formula, the birthday attribute is composed of the birth-year, the birth-month, and the birth-day, and one birthday is expressed as (birth-year $\wedge$ birth-month $\wedge$ birth-day). For the above example, the monotone formula is $(Australia \vee \ldots) \wedge (1915 \vee \ldots \vee (1997 \wedge (Jan. \vee \ldots \vee (Sept. \wedge (1^{st} \vee \ldots \vee 5^{th})))))$. Using this type of formula, the number of public parameters multiplied in the accumulator is decreased to 198 attributes in total of 101 nationalities, 83 birth-years, 9 birth-months, and 5 birth-days, which greatly impacts the reduction of authentication time.

However, the proposed scheme has a drawback against [23]: The size of the user's certificate becomes exponential in the number of the user's attributes, compared to the constant size in [23] (the exponential size can be shortened as shown later). Since the scheme of [23] supports the CNF formula, by converting a monotone formula to a CNF formula, we can employ the scheme of [23] to prove the monotone formula without the drawback of the long certificate. However, in general, the size (the number of literals) of the converted CNF formula becomes super-polynomial in the size of the original monotone formula, which causes a huge authentication time (because the literals in the proved formula are multiplied). On the other hand, by the trade-off of the long certificate, our proposed scheme enables the direct proof of monotone formula. As shown in the above example, it leads the efficient proof generation in the original size of the expressive monotone formula, which is our contribution.

Our approach to prove the monotone formula is that the tag assignment in the accumulator is extended to be adapted to the tree expressing the monotone formula. In the tree, leaves indicate the attributes and internal nodes are the AND or OR relations. For instance, consider the following proved monotone formula:

$$((a_1 \wedge a_2) \vee a_3) \wedge (a_4 \vee a_5) \wedge a_6.$$

As the preparation of the accumulator, a tag assignment is executed as follows. At the root, a series of tags $c_1, \ldots, c_4$ are generated. Then, these tags are divided and assigned to the leaves. The same tags are distributed to the children on an OR relation, while different tags are distributed to the children on an AND relation. The tag assignment result for the above formula is

$$((a_1^{c_1} \wedge a_2^{c_2}) \vee a_3^{c_1,c_2}) \wedge (a_4^{c_3} \vee a_5^{c_3}) \wedge a_6^{c_4},$$

where the superscript in each attribute means the assigned tags. In this assignment, the attributes of the user satisfy the formula if and only if the tags for the attributes are exactly the same as the initial tags. For example, when a user with satisfying attributes a_3, a_5, a_6, the assigned tags are $\{\{c_1, c_2\}, c_3, c_4\}$, which compose the initial tags. In the verification of the accumulator, it is checked using a pairing relation, which is extended from that of [23].

3.2 Extended Accumulator

In this section, we show an accumulator to verify monotone formulas as the key primitive. It is an accumulator extended from the previous accumulator [23]. The extended accumulator compresses the more general formula, the monotone formula, than CNF of the previous [23]. The construction of the accumulator is based on [23], and it employs our new tag assignment algorithm, where tags are assigned in leaf attributes in the binary tree of the given monotone formula.

The difference of constructions between the proposed scheme and [23] is described in *intuition behind construction* in Section 3.2.5.

3.2.1 Notations and Assumptions

In the accumulator, each attribute is indexed by an integer in $\{1, \ldots, n\}$. The set of all attributes has to be fixed in advance, i.e., the small universe (The comparison of this restriction to the previous scheme [23] is shown in Section 3.5.2). Each user owns attributes, and *user's attribute set* U denotes the set of the indices of the attributes that the user owns.

A monotone formula $\mathcal{M}$ is represented by a binary tree, where any internal node is either AND or OR, and the leaf nodes are attributes. For monotone formula $\mathcal{M}$, let $\mathcal{M}_{\mathcal{A}}$ be the set of attribute indices in $\mathcal{M}$. In a monotone formula, the same attribute may be included twice or more, such as $(Japanese \wedge student) \vee (Japanese \wedge professor)$. In this paper, for simplicity, the same attributes in the formula are indexed by different indices (e.g., the first *Japanese* is indexed by 1 and the second one is indexed by 2), while the user's attribute set includes all indices for the attribute. This means that the number of the indices used as the same attribute is fixed in advance (The comparison of this restriction to the previous scheme [23] is also shown in Section 3.5.2). Therefore, we can assume that the attribute indices in $\mathcal{M}$ are all different.

From the tree of a given monotone formula $\mathcal{M}$, consider a *minimal satisfaction tree*, as follows. In any intermediate OR node, one child node remains (because it is needed in minimal for satisfying the OR node), but another redundant child node (and the descendant subtree) is removed. Note that, in any internal AND node, both child nodes remain because the child nodes are needed for the satisfaction of $\mathcal{M}$. Consider user's attribute set U, and $\tilde{U}$ that is a subset of $U \cap \mathcal{M}_{\mathcal{A}}$. We define a predicate $\mathcal{MS}(\tilde{U}, \mathcal{M})$, where $\mathcal{MS}(\tilde{U}, \mathcal{M}) = 1$ if $\tilde{U}$ consists of attributes in a minimal satisfaction tree of $\mathcal{M}$, and otherwise $\mathcal{MS}(\tilde{U}, \mathcal{M}) = 0$. We call set $\tilde{U}$ s.t. $\mathcal{MS}(\tilde{U}, \mathcal{M}) = 1$ a *minimal attribute set* of $\mathcal{M}$ and we denote such $\tilde{U}$

as $\hat{U}$. In the example of Fig. 3.1, the sub-tree connected by the double lines is a minimal satisfaction tree with the minimal attribute set $\hat{U} = \{a_3, a_5, a_6\}$. Here, note that minimal satisfaction tree is not unique. In the example of Fig.1, we can consider another minimal satisfaction tree for another minimum attribute set $\hat{U} = \{a_3, a_4, a_6\}$.

We assume that $|U|$ for every user is bounded by the upper bound η. The value η is fixed by depending on $|U|$ of all users, but does not depend on proved formulas. The value of η must be fixed before the setup. Thus, in the application such as the proposed anonymous credential system, some authority needs to estimate the value of η from the number of attributes that every user can use. This is a restriction, which we compare to the previous system [23] in Section 3.5.2.

3.2.2 Syntax and Security of Extended Accumulator

We show the definition of algorithms in the extended accumulator as follows:

AccSetup: This is the algorithm to output the public parameters pk_{acc} and only be executed once.

AccGen: This is the algorithm to compute an accumulator, i.e., the compressed value of a given formula. Given pk_{acc} and a monotone formula $\mathcal{M}$, this algorithm outputs the accumulator, $acc_{\mathcal{M}}$, together with the auxiliary values $aux_{\mathcal{M}}$.

AccWitGen: This is the algorithm to compute the witness W of the minimal satisfaction of $\mathcal{M}$ by $\hat{U}$ (i.e., $\mathcal{MS}(\hat{U}, \mathcal{M}) = 1$). Given pk_{acc}, $\hat{U}, \mathcal{M}$, $aux_{\mathcal{M}}$, this algorithm outputs W.

AccVerify: This is the algorithm to verify the minimal satisfaction of $\mathcal{M}$ by $\hat{U}$. Given pk_{acc}, $acc_{\mathcal{M}}, \hat{U}, W$, $aux_{\mathcal{M}}$, this algorithm accepts them if $\mathcal{MS}(\hat{U}, \mathcal{M}) = 1$ and reject them if otherwise.

We define the correctness and the security of the extended accumulator by the folowing requirements.

Correctness : The extended accumulator is *correct* if **AccSetup** algorithm correctly computes pk_{acc} and **AccGen** and **AccWitGen** correctly output $acc_{\mathcal{M}}$, $aux_{\mathcal{M}}$, and W for $\hat{U}$ and $\mathcal{M}$, then **AccVerify** accepts $acc_{\mathcal{M}}, \hat{U}, W$ that are output by the algorithms, if $\mathcal{MS}(\hat{U}, \mathcal{M}) = 1$.

Security : Consider the following game between a challenger and the adversary:

Game$_{\textbf{Acc}}$:

1. The challenger runs the **AccSetup** algorithm to generate the public parameters pk_{acc}. It gives pk_{acc} to the adversary.
2. The adversary outputs $\tilde{U}, \mathcal{M}$ and W.

Then, the adversary wins if

- For $acc_{\mathcal{M}}$, $aux_{\mathcal{M}}$ which are the outputs of **AccGen** given pk_{acc} and $\mathcal{M}$, **AccVerify** accepts pk_{acc}, $acc_{\mathcal{M}}, \tilde{U}$, W, $aux_{\mathcal{M}}$, but

- $\mathcal{MS}(\tilde{U}, \mathcal{M}) = 0$.

Then, the extended accumulator is *secure* if any PPT adversary can win $\mathbf{Game_{Acc}}$ only with negligible probability.

3.2.3 Tag Assignment Algorithm

In this tag assignment algorithm, the input is a monotone formula $\mathcal{M}$ on attributes. The output of the algorithm is a non-negative integer T showing the number of tag indices, and the sequence of tag indices, $\mathcal{S}_i = [\mathtt{a}..\mathtt{b}]$ to each attribute i in $\mathcal{M}$, where $[\mathtt{a}..\mathtt{b}]$ denotes the set of consecutive integers between $\mathtt{a}$ and $\mathtt{b}$, i.e., $\{\mathtt{a}, \mathtt{a}+1, \ldots, \mathtt{b}\}$.

The goal of this algorithm is to output a partition $\{\mathcal{S}_i\}_{i \in \hat{U}}$ of the initial set $\mathcal{S}_\epsilon$=$[1..T]$, for the minimal attribute set $\hat{U}$ of $\mathcal{M}$, which is used for verifying the monotone formula in the accumulator.

Intuition behind construction: In the algorithm, each node is traversed from the root node as follows. At an AND node, the sequences of tag's indices is separated and given to each child node. This is because the combination of the sequences from both subtrees rooted by the two child nodes can be equal to the AND node's sequence. At an OR node, the sequence of tag's indices is exactly given to both child nodes. This is why the sequence of the subtree rooted by either of the child nodes can become the sequence of the OR node. To ensure the separation at descendant's AND nodes, an auxiliary sequence $\mathcal{A}$ is introduced. $\mathcal{A}$ is a sequence of separable positions in the sequence of tag's indices. When traversing the tree, $\mathcal{A}$ is separated and assigned to each child node such that the number of separable positions is equal to the number of AND nodes in the subtree rooted at the child node. Note that, at an AND node, one separable position is consumed for the separation of tag's indices.

Preparation: For every node $\mathtt{N}$, traverse the tree to find $T_{\mathtt{N}}$ that is the number of AND nodes in the subtree rooted by $\mathtt{N}$. At the root node $\mathtt{N}$=ϵ, the number of AND nodes is set as T_ϵ. Then, set $T = T_\epsilon + 1$, the root's sequence $\mathcal{S}_\epsilon$=$[1..T]$, and the auxiliary sequence $\mathcal{A}_\epsilon$=$[1..T_\epsilon]$ that includes the separable positions of $\mathcal{S}_\epsilon$.

Assignment: Using the following function ASSIGN(N, $\mathcal{S}$,$\mathcal{A}$) where N is a node, $\mathcal{S}$=$[\mathtt{a}..\mathtt{b}]$, and $\mathcal{A}$=$[\mathtt{c}..\mathtt{d}]$, assign $\mathcal{S}$ and $\mathcal{A}$ to each node recursively starting with ASSIGN(ϵ,$\mathcal{S}_\epsilon$,$\mathcal{A}_\epsilon$). Then, output the sequence $\mathcal{S}_i$ of every leaf node for attributes $i \in \mathcal{M}_{\mathcal{A}}$. The function ASSIGN(N,$\mathcal{S}$,$\mathcal{A}$) is defined as follows. Here, LeftChild(N) (resp., RightChild(N)) denotes as the left (resp., right) child node of N.

<u>ASSIGN(N,$\mathcal{S}$,$\mathcal{A}$)</u>
if N is an OR node:

1. Split $\mathcal{A}$ into $\mathcal{A}'$=$[\mathtt{c}..\mathtt{c}+T_{\mathtt{LeftChild(N)}}-1]$ and $\mathcal{A}''$=$[\mathtt{c}+ T_{\mathtt{LeftChild(N)}}..\mathtt{d}]$, where $\mathcal{A}'$ (resp., $\mathcal{A}''$) is set as empty, if $T_{\mathtt{LeftChild(N)}}$=0 (resp., $T_{\mathtt{RightChild(N)}}$=0).
2. Run ASSIGN(LeftChild(N),$\mathcal{S}$,$\mathcal{A}'$).
3. Run ASSIGN(RightChild(N),$\mathcal{S}$,$\mathcal{A}''$).
4. Return.

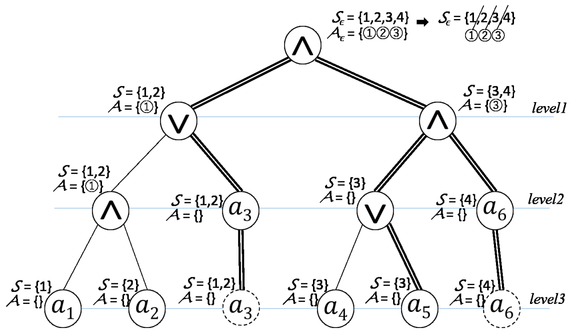

Figure 3.1: Tag assignment in the tree of a monotone formula.

if N is an AND node:

1. Split $\mathcal{A}$ into $\mathcal{A}'$=[c..c+$T_{\texttt{LeftChild(N)}}$-1] and $\mathcal{A}''$=[c+ $T_{\texttt{LeftChild(N)}}$+1..d], where $\mathcal{A}'$ (resp., $\mathcal{A}''$) is set as empty, if $T_{\texttt{LeftChild(N)}}$=0 (resp., $T_{\texttt{RightChild(N)}}$=0). In this case, one separable position a+ $T_{\texttt{LeftChild(N)}}$ is consumed in $\mathcal{A}$.
2. Using the separate position $\texttt{a} + T_{\texttt{LeftChild(N)}}$, split $\mathcal{S}'$ into $\mathcal{S}'$=[a..c+$T_{\texttt{LeftChild(N)}}$] and $\mathcal{S}''$ = [c+$T_{\texttt{LeftChild(N)}}$+1..b].
3. Run ASSIGN(LeftChild(N),$\mathcal{S}'$,$\mathcal{A}'$).
4. Run ASSIGN(RightChild(N),$\mathcal{S}''$,$\mathcal{A}''$).
5. Return.

if N is a leaf node for attribute i:

1. Output $\mathcal{S}_i = \mathcal{S}$ as the sequence of the attribute i.
2. Return.

We explain the algorithm using an example of a monotone formula.
Example. Suppose we are given a monotone formula $((a_1 \wedge a_2) \vee a_3) \wedge (a_4 \vee a_5) \wedge a_6$ whose tree is shown in Fig. 3.1.

In Fig. 3.1, $\mathcal{S}_\epsilon = \{1, 2, 3, 4\}$ and $\mathcal{A}_\epsilon$={1, 2, 3}. The separable position t in $\mathcal{A}$ means that $\mathcal{S}$= {a, . . . , t, . . . , b} is separable to $\mathcal{S}'$ = [a..t] and $\mathcal{S}''$= [t+1..b]. When traversing the AND node, one separable position in $\mathcal{A}$ is consumed, and $\mathcal{S}$ is divided into the left and right child nodes. The remaining separable positions are also distributed to the child nodes such that the descendant AND nodes can consume the separable positions. In Fig. 3.1, the root node is an AND node and it consumes separable positions ②, thus $\mathcal{S}' = \{1, 2\}$ and $\mathcal{S}'' = \{3, 4\}$ are assigned to left child and right child, respectively. The separable positions $\mathcal{A}$ = {①, ②, ③} are distributed to $\mathcal{A}'$ = {①} and $\mathcal{A}''$ = {③}, where ② is consumed in the root.

When the current node is an OR node, none of the separable position is consumed and thus both child nodes receive the same $\mathcal{S}$. The separable positions are distributed to the child nodes such that the descendant AND nodes can consume the separable position. In the OR node of level 1 (Fig. 3.1), the separable position ① is distributed to the left child with the AND descendant.

In Fig. 3.1, the subtree connected with double lines branch is the minimal satisfaction tree where $\hat{U} = \{a_3, a_5, a_6\}$. The set of $\mathcal{S}_i$ for all $i \in \hat{U}$, i.e., $\{\{1,2\},\{3\},\{4\}\}$ is a partition of $\mathcal{S}_\epsilon = \{1,2,3,4\}$.

3.2.4 Correctness of Tag Assignment Algorithm

We have the following theorem of the correctness of the tag assignment algorithm.

Theorem 1. *For $\mathcal{S}_i$ and $\mathcal{S}_\epsilon$ output in the tag assignment algorithm on input $\mathcal{M}$, for any set of attributes $\tilde{U}$, if and only if $\mathcal{MS}(\tilde{U}, \mathcal{M}) = 1$, $\{\mathcal{S}_i | i \in \tilde{U}\}$ is a partition of $\mathcal{S}_\epsilon$, i.e., $\bigcup_{i \in \tilde{U}} \mathcal{S}_i = \mathcal{S}_\epsilon$ and $\mathcal{S}_i$'s are mutually disjoint.*

Proof. First, let us consider the proof that, if $\mathcal{MS}(\tilde{U}, \mathcal{M}) = 1$, $\{\mathcal{S}_i | i \in \tilde{U}\}$ is a partition of $\mathcal{S}_\epsilon$. In the algorithm, since each $T_{\mathtt{N}}$ is correctly computed, each AND node can always consume a separable position in $\mathcal{A}$. Note that, at each level in the minimal satisfaction tree, the set of all $\mathcal{S}$ of nodes at the level remains a partition of $\mathcal{S}_\epsilon$. This is because $\mathcal{S}$ in an AND node is partitioned to two child nodes and $\mathcal{S}$ in an OR node is inherited to the child in the minimal satisfaction. As in Fig. 3.1, a leaf node at a level is replaced by a virtual intermediate node connected to the virtual leaf node at the bottom level via the virtual intermediate nodes. Then, at the bottom level, the set of all $\mathcal{S}_i$ remains a partition of $\mathcal{S}_\epsilon$.

Next, consider the reverse proof that, if $\mathcal{MS}(\tilde{U}, \mathcal{M}) \neq 1$, $\{\mathcal{S}_i | i \in \tilde{U}\}$ is not a partition of $\mathcal{S}_\epsilon$. Consider the subtree of $\mathcal{M}$ with only the paths from the root to all leaves in $\tilde{U}$. We assume that $\tilde{U}$ is not empty, because it also means that $\{\mathcal{S}_i | i \in \tilde{U}\}$ is also empty. In this case, since the subtree is not the satisfaction tree, there is an AND node $\mathtt{N}^*$ with `LeftChild(N*)` connected to a leaf in $\tilde{U}$ and `RightChild(N*)` that is not connected to the leaves in $\tilde{U}$ (or `RightChild(N*)` is connected to the leaves and `LeftChild(N*)` is not connected, but this case is omitted since it is similarly discussed.). In this AND node $\mathtt{N}^*$, tag indices are divided to $\mathcal{S}'$=[a..t] and $\mathcal{S}''$=[t+1..b] at a separable position ⓣ. Then, tag indices [t+1..b] are not assigned to leaf attributes in $\tilde{U}$ that are descendants of node $\mathtt{N}^*$. On the other hand, from ancestors of node $\mathtt{N}^*$, using other paths that do not include node $\mathtt{N}^*$, the tag indices [t+1..b] may be distributed to leaf attributes in $\tilde{U}$. However, this is not true. This is because tag indices are not divided at ⓣ in other AND nodes, which means that any leaf except the descendants of $\mathtt{N}^*$ cannot obtain [t+1..b′] for any b′. Therefore, $\{\mathcal{S}_i | i \in \tilde{U}\}$ is not a partition of $\mathcal{S}_\epsilon$. □

3.2.5 Accumulator to Verify Monotone Formulas

The following accumulator is used to verify a monotone formula where the formula is compressed into one single value to obtain a constant-size proof for the number of all attributes of a user and the size of the proved formula.

AccSetup: This is the algorithm to output the public parameters. Select bilinear group $\mathbb{G}_1$ and $\mathbb{G}_2$ with a prime order p and a bilinear map e. Select $\gamma \in_R \mathbb{Z}_p$, and compute and publish $p, \mathbb{G}_1, \mathbb{G}_2, e, g, g_1 = g^{\gamma^1}, \dots, g_n = g^{\gamma^n}, g_{n+2} = g^{\gamma^{n+2}}, \dots, g_{2n} = g^{\gamma^{2n}}, \tilde{g}, \tilde{g}_1 = \tilde{g}^{\gamma^1}, \dots, \tilde{g}_n = \tilde{g}^{\gamma^n}, \tilde{g}_{n+2} = \tilde{g}^{\gamma^{n+2}}, \dots, \tilde{g}_{2n} = \tilde{g}^{\gamma^{2n}}$ and $z = e(g, \tilde{g})^{\gamma^{n+1}}$ as the public parameters.

AccGen: This is the algorithm to compute the accumulator given the public parameters and a monotone formula $\mathcal{M}$. For input $\mathcal{M}$, run the tag assignment algorithm. The tag assignment algorithm outputs T and $\mathcal{S}_i$ for $i \in \mathcal{M}_{\mathcal{A}}$, where T is the total number of tag indices and $\mathcal{S}_i$ is the set of tag indices assigned to each attribute i. Set a tag value as $c_t = (\eta+1)^{t-1}$ for all $1 \le t \le T$. We assume that $(\eta+1)c_T < p$. The accumulator of $\mathcal{M}$ outputs $acc_{\mathcal{M}} = \prod_{i\in\mathcal{M}_{\mathcal{A}}} g_{n+1-i}^{\sum_{t\in\mathcal{S}_i} c_t}$, together with $\mathcal{S}_1, \ldots, \mathcal{S}_{|\mathcal{M}_{\mathcal{A}}|}$, $c_1, \ldots, c_T$.

AccWitGen: This is the algorithm to compute the witness of the minimal satisfaction of $\mathcal{M}$ by $\hat{U}$ (i.e.,$\hat{U}$ s.t. $\mathcal{MS}(\hat{U}, \mathcal{M}) = 1$), given the public parameters and $\hat{U}, \mathcal{M}$, $\mathcal{S}_1, \ldots, \mathcal{S}_{|\mathcal{M}_{\mathcal{A}}|}$, $c_1, \ldots, c_T$. The witness is computed as $W = \prod_{j\in\hat{U}} \prod_{i\in\mathcal{M}_{\mathcal{A}}, i\neq j} \tilde{g}_{n+1-i+j}^{\sum_{t\in\mathcal{S}_i} c_t}$.

AccVerify: This is the algorithm to verify the minimal satisfaction of $\mathcal{M}$ by $\hat{U}$, given the public parameters and $acc_{\mathcal{M}}, \hat{U}, W$, $c_1, \ldots, c_T$. Set $u = c_1 + \ldots + c_T$, accept if,

$$\frac{e(acc_{\mathcal{M}}, \prod_{i\in\hat{U}} \tilde{g}_i)}{e(g, W)} = z^u.$$

Intuition behind construction: Each algorithm in the extended accumulator is basically the same as the underlying scheme [23] for CNF formulas. The only difference is how to utilize tags. In [23], each attribute is assigned to index i, and a tag c_k is assigned to the k-th OR clause in the CNF formula. The calculation of acc is the multiplications of $g_{n+1-i}^{c_k}$ for all attributes i in the CNF formula. In this case, in this accumulator's verification equation, the left-hand side produces z^{c_k} for a matched attribute from the user's attribute set U. Therefore, if U includes an attribute in every clause of CNF formula, $z^{c_1+\cdots+c_T}$ is produced in the left-hand and the verification equation holds.

In the extended scheme, a tree expression of the monotone formula is constructed, and a sequence of tags is assigned to each attribute. Then, as an output of the tag assignment algorithm, if $\hat{U}$ is a minimal attribute set, the tags of attributes in $\hat{U}$ are exactly $c_1, \ldots, c_T$. The computation of acc in the extended scheme is similar to [23], which is the multiplications of $g_{n+1-i}^{\sum_{t\in\mathcal{S}_i} c_t}$ for every attribute i in $\mathcal{M}$, where $\mathcal{S}_i$ is the set of tags assigned to attribute i. The verification equation is the same as in [23]. Thus, due to the same principle, the left-hand side produces $z^{\sum_{t\in\mathcal{S}_i} c_t}$ for a matched attribute i from $\hat{U}$. Therefore, if $\hat{U}$ is a minimal attribute set, $z^{\sum_{i\in\hat{U}} \sum_{t\in\mathcal{S}_i} c_t} = z^{c_1+\cdots+c_T}$ is produced in the left-hand, and the verification equation holds.

3.2.6 Correctness and Security of Accumulator

Based on Section 3.2.2, we prove that the proposed accumulator is *correct* and *secure*, as follows.

Theorem 2. *The proposed accumulator is correct.*

Proof. By substituting $acc_{\mathcal{M}}$ and W to the verification equation, the left hand is equal to

$$\begin{aligned}
& \frac{e(\prod_{i\in\mathcal{M}_{\mathcal{A}}} g_{n+1-i}^{\sum_{t\in\mathcal{S}_i} c_t}, \prod_{j\in\hat{U}} \tilde{g}_j)}{e(g, \prod_{j\in\hat{U}} \prod_{i\in\mathcal{M}_{\mathcal{A}}, i\neq j} \tilde{g}_{n+1-i+j}^{\sum_{t\in\mathcal{S}_i} c_t})} \\
= & \frac{e(g, \prod_{j\in\hat{U}} \prod_{i\in\mathcal{M}_{\mathcal{A}}} \tilde{g}_{n+1-i+j}^{\sum_{t\in\mathcal{S}_i} c_t})}{e(g, \prod_{j\in\hat{U}} \prod_{i\in\mathcal{M}_{\mathcal{A}}, i\neq j} \tilde{g}_{n+1-i+j}^{\sum_{t\in\mathcal{S}_i} c_t})} \\
= & \; e(g, \prod_{i\in\hat{U}} \tilde{g}_{n+1}^{\sum_{t\in\mathcal{S}_i} c_t}) = e(g, \tilde{g}_{n+1}^{\sum_{i\in\hat{U}} \sum_{t\in\mathcal{S}_i} c_t}).
\end{aligned}$$

This is equal to $z^u = e(g, \tilde{g}_{n+1})^{c_1+\cdots+c_T}$, since $\{\mathcal{S}_i | i \in \hat{U}\}$ is a partition of $\mathcal{S}_\epsilon$ due to Theorem 1. □

For proving the security of the accumulator, we prepare the following lemma.

Lemma 1. *For any $\bar{t}$ $(2 \leq \bar{t} \leq T)$, $c_{\bar{t}} > \sum_{1\leq t \leq \bar{t}-1} \eta \cdot c_t$.*

Proof. In the case of $\bar{t} = 2$, $c_2 = (\eta + 1) \cdot c_1 > \eta \cdot c_1$. For $\bar{t} \geq 3$, we assume the case of $\bar{t} - 1$, that is $c_{\bar{t}-1} > \sum_{1\leq t\leq \bar{t}-2} \eta \cdot c_t$, and we will prove the case of $\bar{t}$. Using the assumption and $c_{\bar{t}} = (\eta + 1) \cdot c_{\bar{t}-1}$, we have

$$\begin{aligned}
&\sum_{1\leq t\leq \bar{t}-1} \eta \cdot c_t \\
= \;& \eta \cdot c_{\bar{t}-1} + \sum_{1\leq t\leq \bar{t}-2} \eta \cdot c_t \\
< \;& \eta \cdot c_{\bar{t}-1} + c_{\bar{t}-1} \\
= \;& (\eta + 1) \cdot c_{\bar{t}-1} \\
= \;& c_{\bar{t}}
\end{aligned}$$

Thus, for any $\bar{t}$ $(2 \leq \bar{t} \leq L)$, we obtain $c_{\bar{t}} > \sum_{1\leq t\leq \bar{t}-1} \eta \cdot c_t$. □

Theorem 3. *Under the n-DHE assumption, the proposed accumulator is secure.*

Proof. Assume the existence of such an adversary that outputs the described elements with non negligible probability. Let $\tilde{g}_{n+1} = \tilde{g}^{\gamma^{n+1}}$. By substituting $acc_{\mathcal{M}}$ to the calculation in the verification equation, we can obtain

$$\begin{aligned}
\frac{e(\prod_{i\in\mathcal{M}_\mathcal{A}} g_{n+1-i}^{\sum_{t\in\mathcal{S}_i} c_t}, \prod_{j\in\hat{U}} \tilde{g}_j)}{e(g, W)} &= z^u = e(g, \tilde{g}_{n+1})^u, \\
e(g, \prod_{j\in\tilde{U}} \prod_{i\in\mathcal{M}_\mathcal{A}} \tilde{g}_{n+1-i+j}^{\sum_{t\in\mathcal{S}_i} c_t}) &= e(g, W \cdot \tilde{g}_{n+1}^u),
\end{aligned}$$

where $\mathcal{S}_1,\ldots,\mathcal{S}_{\mathcal{M}_\mathcal{A}}$ are the output of **AccGen** for $\mathcal{M}$. Thus, we have

$$\prod_{j\in\tilde{U}} \prod_{i\in\mathcal{M}_\mathcal{A}} \tilde{g}_{n+1-i+j}^{\sum_{t\in\mathcal{S}_i} c_t} = W \cdot \tilde{g}_{n+1}^u.$$

Here, for all $1 \leq t \leq T$, let λ_t be the number of $\mathcal{S}_i$ s.t. $t \in \mathcal{S}_i$ for $i \in \tilde{U}$. In this case, note that $\lambda_t \leq \eta$, since $\lambda_t \leq |\tilde{U}|$ and η is the upper bound of $|\tilde{U}|$. Then, we have

$$\prod_{j\in\tilde{U}} \prod_{i\in\mathcal{M}_\mathcal{A}, i\neq j} \tilde{g}_{n+1-i+j}^{\sum_{t\in\mathcal{S}_i} c_t} \cdot \prod_{1\leq t\leq T} \tilde{g}_{n+1}^{\lambda_t c_t} = W \cdot \tilde{g}_{n+1}^u$$

$$\prod_{j\in\tilde{U}} \prod_{i\in\mathcal{M}_\mathcal{A}, i\neq j} \tilde{g}_{n+1-i+j}^{\sum_{t\in\mathcal{S}_i} c_t} = W \cdot \tilde{g}_{n+1}^{u-\sum_{1\leq t\leq T} \lambda_t c_t} \tag{3.1}$$

Set $\Delta = u - \sum_{1\leq t\leq T} \lambda_t c_t = \sum_{1\leq t\leq T} (1 - \lambda_t) c_t$, due to $u = c_1 + \ldots + c_T$.

Equation (3.1) means that, if $\Delta \neq 0 \pmod p$, we can compute $\tilde{g}_{n+1}$ from the other $\tilde{g}_1, \ldots, \tilde{g}_n, \tilde{g}_{n+2}, \ldots, \tilde{g}_{2n}$. In the following, we prove $\Delta \neq 0 \pmod p$.

Separate $\mathcal{S}_{root} = \{1, \ldots, T\}$ to $\mathcal{T}^>$, $\mathcal{T}^<$ and $\mathcal{T}^=$, where $\mathcal{T}^>$ consists of t s.t. $(1-\lambda_t) > 0$, $\mathcal{T}^<$ consists of t s.t. $(1-\lambda_t) < 0$, and $\mathcal{T}^=$ consists of t s.t. $(1-\lambda_t) = 0$. We can obtain

$$\begin{aligned}\Delta &= \sum_{t\in\mathcal{T}^>}(1-\lambda_t)c_t + \sum_{t\in\mathcal{T}^<}(1-\lambda_t)c_t + \sum_{t\in\mathcal{T}^=}(1-\lambda_t)c_t \\ &= \sum_{t\in\mathcal{T}^>}(1-\lambda_t)c_t + \sum_{t\in\mathcal{T}^<}(1-\lambda_t)c_t,\end{aligned}$$

due to $1-\lambda_t = 0$ for all $t \in \mathcal{T}^=$.

Let $\tilde{t}$ be the maximum of t s.t. $t \notin \mathcal{T}^=$ (i.e., $\tilde{t} \in \mathcal{T}^>$ or $\tilde{t} \in \mathcal{T}^<$). From the assumption of $\mathcal{MS}(\tilde{U}, \mathcal{M}) = 0, \{\mathcal{S}_i | i \in \tilde{U}\}$ is not a partition of $\mathcal{S}_\epsilon$ and thus $\lambda_t \neq 1$ for some t, which ensures the existence of such $\tilde{t}$. Consider two cases.

(i) The first case is that $\tilde{t} \in \mathcal{T}^<$ (i.e., $1 < \lambda_{\tilde{t}}$). Then, $(1-\lambda_{\tilde{t}})c_{\tilde{t}} \leq -c_{\tilde{t}}$. This is why

$$\Delta \leq -c_{\tilde{t}} + \sum_{t\in\mathcal{T}^>}(1-\lambda_t)c_t + \sum_{t\in\mathcal{T}^<, t\neq\tilde{t}}(1-\lambda_t)c_t.$$

For $t \in \mathcal{T}^>$, $1-\lambda_t > 0$. However, due to $\lambda_t \geq 0$, we have $\lambda_t = 0$, i.e., $1-\lambda_t = 1$. For $t \in \mathcal{T}^<$, we have $1-\lambda_t < 0$. Thus,

$$\Delta < -c_{\tilde{t}} + \sum_{t\in\mathcal{T}^>} c_t.$$

From Lemma 1, we have $c_{\tilde{t}} > \sum_{t\in\mathcal{T}^>} c_t$, due to $\tilde{t} > t$ for any $t \in \mathcal{T}^>$. Thus, $-c_{\tilde{t}} + \sum_{t\in\mathcal{T}^>} c_t < 0$. Therefore, we can obtain $\Delta < 0$. On the other hand, we obtain

$$\Delta = \sum_{1\leq t\leq T}(1-\lambda_t)c_t > -\sum_{1\leq t\leq T}\eta c_t,$$

due to $1-\lambda_t > -\eta$ which is derived from $\lambda_t \leq \eta$. From Lemma 1, we have $\sum_{1\leq t\leq T-1,}\eta c_t < c_T$, and thus

$$\sum_{1\leq t\leq T}\eta c_t < c_T + \eta c_T = (\eta+1)c_T < p.$$

Thus, $\Delta > -p$. Therefore, we have $\Delta \neq 0 \pmod p$.

(ii) The other case is that $\tilde{t} \in \mathcal{T}^>$ (i.e., $1 > \lambda_{\tilde{t}}$). Then, due to $(1-\lambda_{\tilde{t}})c_{\tilde{t}} \geq c_{\tilde{t}}$, we obtain

$$\Delta \geq c_{\tilde{t}} + \sum_{t\in\mathcal{T}^>, t\neq\tilde{t}}(1-\lambda_t)c_t + \sum_{t\in\mathcal{T}^<}(1-\lambda_t)c_t.$$

For any $t \in \mathcal{T}^>$, (i.e., $1-\lambda_t > 0$), $(1-\lambda_t)c_t > 0$ and thus

$$\Delta > c_{\tilde{t}} + \sum_{t\in\mathcal{T}^<}(1-\lambda_t)c_t.$$

Due to $\tilde{t} > t$ for any $t \in \mathcal{T}^<$, from Lemma 1 and $\lambda_t - 1 < \eta$,

$$c_{\tilde{t}} > \sum_{t \in \mathcal{T}^<} \eta c_t > \sum_{t \in \mathcal{T}^<} (\lambda_t - 1) c_t,$$

and thus

$$c_{\tilde{t}} + \sum_{t \in \mathcal{T}^<} (1 - \lambda_t) c_t > 0.$$

This is why we obtain $\Delta > 0$. On the other hand, from $1 - \lambda_t < 1$ and Lemma 1,

$$\Delta \leq \sum_{1 \leq t \leq T} c_t = \sum_{1 \leq t \leq T-1} c_t + c_T \leq \ c_T + c_T.$$

Thus, $\Delta \leq 2c_T < p$. Therefore, in this case, also $\Delta \neq 0 \pmod{p}$.

From equation (1),

$$\tilde{g}_{n+1} = (W^{-1} \prod_{j \in \tilde{U}} \prod_{i \in \mathcal{M}_\mathcal{A}, i \neq j} \tilde{g}_{n+1-i+j}^{\sum_{t \in \mathcal{S}_i} c_t})^{1/(u - \sum_{1 \leq t \leq T} \lambda_t c_t)}$$

For any $j \in \tilde{U}$ and any $i \in \mathcal{M}_\mathcal{A}$ satisfying $i \neq j$, we have $\tilde{g}_{n+1-i+j} \neq \tilde{g}_{n+1}$, and $\Delta = u - \sum_{1 \leq t \leq T} \lambda_t c_t \neq 0$. Thus, we can compute $\tilde{g}_{n+1}$ with non-negligible probability given $\tilde{g}_1, \ldots, \tilde{g}_n, \tilde{g}_{n+2}, \ldots, \tilde{g}_{2n}$, which contradicts n-DHE assumption. $\square$

3.3 Syntax and Security Model of Anonymous Credential System

The syntax and security model of an anonymous credential system for monotone formulas can be defined similarly to the previous work [23]. Here, we show the syntax and the security model.

3.3.1 Syntax

The attribute value is indexed by an integer from $\{1, \ldots, n\}$, where n is the total number of attribute values. All attribute values in all attribute types are indexed by using the universal set $\{1, \ldots, n\}$.

The anonymous credential system consists of the following algorithms:

IssuerKeyGen: The inputs of this algorithm are n, η, where η is the maximum number of users' attributes. The outputs are issuer's public key ipk and issuer's secret key isk.

CertObtain: This is an interactive protocol between a probabilistic algorithm **CertObtain-**$\mathcal{U}$ for the user and a probabilistic algorithm **CertObtain-**$\mathcal{I}$ for an issuer, where the issuer issues the certificate including the attributes to the user. **CertObtain-**$\mathcal{U}$, on input ipk and $U \subset \{1, \ldots, n\}$ that is indices corresponding to the attributes of the user, outputs the certificate $cert$ ensuring the attributes of the user. On the other hand, **CertObtain-**$\mathcal{I}$ is given ipk, isk as inputs.

ProofGen: This probabilistic algorithm, on inputs ipk, U, $cert$, $\mathcal{M}$ that is the monotone formula on attributes to be proved, outputs the proof σ.

Verify: This is a deterministic algorithm for verification. The input is ipk, a proof σ, and the formula $\mathcal{M}$. Then the output is 'valid' if the attributes in U satisfy $\mathcal{M}$, or 'invalid' otherwise.

3.3.2 Security Model

The security model consists of misauthentication resistance and anonymity. The misauthentication resistance requirement captures the soundness of the attribute proof. This means that an adversary $\mathcal{A}$ cannot try to forge a proof for a monotone formula, where the attributes of any user corrupted by $\mathcal{A}$ do not satisfy the formula. The anonymity requirement captures the anonymity and unlinkability of proofs, as in the group signatures.

3.3.2.1 Misauthentication Resistance

Consider the following misauthentication resistance game.

Misauthentication Resistance Game: The challenger runs **IssuerKeyGen**, and obtains ipk and isk. He provides $\mathcal{A}$ with ipk, and run $\mathcal{A}$. He sets CU with empty, where CU denotes the set of IDs of users corrupted by $\mathcal{A}$. In the run, $\mathcal{A}$ can query the challenger about the following issuing query:

C-Issuing: $\mathcal{A}$ can request the certificates on attribute set $U^{(\mathsf{i})}$ of user i. Then, $\mathcal{A}$ as the user executes **CertObtain** protocol with the challenger as the issuer.

Finally, $\mathcal{A}$ outputs a monotone formula $\mathcal{M}^*$, and a proof σ^*.

Then, $\mathcal{A}$ wins if

1. **Verify**$(ipk, \sigma^*, \mathcal{M}^*) =$ valid, and
2. for all $\mathsf{i} \in CU$, $U^{(\mathsf{i})}$ does not satisfy $\mathcal{M}^*$.

Misauthentication resistance requires that for all PPT $\mathcal{A}$, the probability that $\mathcal{A}$ wins the misauthentication resistance game is negligible.

3.3.2.2 Anonymity

Consider the following anonymity game.

Anonymity Game: The challenger runs **IssuerKeyGen**, and obtains ipk, isk. He provides $\mathcal{A}$ with ipk, isk, and run $\mathcal{A}$. He sets HU with empty. In the run, $\mathcal{A}$ can query the challenger, as follows.

H-Issuing: $\mathcal{A}$ can request the certificates on attribute set $U^{(\mathsf{i})}$ of user i. Then, $\mathcal{A}$ as the issuer executes **CertObtain** protocol with the challenger as the user. The challenger adds this user to HU.

Proving: $\mathcal{A}$ can request the user i's proof on formula $\mathcal{M}$. Then, the challenger responds the proof on $\mathcal{M}$ of the user i, if the user is in HU.

During the run, as the challenge, $\mathcal{A}$ outputs a formula $\mathcal{M}$, and two users i_0 and i_1, such that both U_{i_0} and U_{i_1} satisfy $\mathcal{M}^*$. If $\mathsf{i}_0 \in HU$ and $\mathsf{i}_1 \in HU$, the challenger chooses $\phi \in_R \{0, 1\}$, and responds the proof on $\mathcal{M}^*$ of user i_ϕ. After that, similarly, $\mathcal{A}$ can make the queries.

Finally, $\mathcal{A}$ outputs a bit ϕ' indicating its guess of ϕ.

If $\phi' = \phi$, $\mathcal{A}$ wins. We define the advantage of $\mathcal{A}$ as $|\Pr[\phi' = \phi] - 1/2|$. Anonymity requires that for all PPT $\mathcal{A}$, the advantage of $\mathcal{A}$ on the anonymity game is negligible.

3.4 Proposed Anonymous Credential System

3.4.1 Construction Overview

Basically, the construction of the proposed system is similar to the previous system [23], as follows. The certificate of attributes is an AHO signature, where the user's attributes in set U are unified to $P_U = \prod_{i \in U} \tilde{g}_i$ and embedded for the accumulator verification. Together with the accumulated value $acc_{\mathcal{M}}$, P_U are applied in the verification equation of the accumulator to authenticate the user. In the authentication, the user proves that the attributes in U satisfy the given monotone formula $\mathcal{M}$, using the verification equation of the accumulator in Section 3.2.5. To conceal any information beyond the satisfaction, we utilize the GS proofs for the pairing equations in the verification of the AHO signature and the accumulator verification.

However, U might not be a minimal attribute set which causes a failure in the verification. Thus, the user must derive $\hat{U}$ from U such that $\hat{U}$ is a minimal attribute set and use $\hat{U}$ in the verification equation. However, it is not easy for the user to generate a certificate of $P_{\hat{U}}$ from the issued certificate of P_U. Hence, we have the following approach: Consider all possible candidates $U_k (1 \leq k \leq K)$ of subsets of U such that a U_k satisfying $\mathcal{MS}(U_k, \mathcal{M}) = 1$ exists for any monotone formula $\mathcal{M}$. Then, the user is issued certificates for all U_k. In the authentication, for a given monotone formula $\mathcal{M}$, the user selects a certificate for a $U_k = \hat{U}$ such that $\hat{U}$ is the minimal attribute set, and can prove the correctness of the certified attributes of the user using the verification of AHO signatures.

Compared to [23], there are two differences in construction: The first one is that the accumulator for CNF formulas in [23] is replaced by our extended accumulator for monotone formulas in Section 3.2.5. The second one is that the certificate on P_U for user's attribute set U is replaced by the certificates on P_{U_k} for all subsets $U_k \subset U$, as shown above.

3.4.2 Construction

IssuerKeyGen. It is given n that is the total number of attributes and η that is the maximum number of user's attributes. Here, η is fixed by the authority in advance.

1. Select bilinear groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ with the same order p and the bilinear map e, and generators $g \in_R \mathbb{G}_1$, $\tilde{g} \in_R \mathbb{G}_2$.
2. Generate public parameters of the accumulator: Select $\gamma \in_R \mathbb{Z}_p$, and compute $pk_{acc} = (g_1 = g^{\gamma^1}, \ldots, g_n = g^{\gamma^n}, g_{n+2} = g^{\gamma^{n+2}}, \ldots, g_{2n} = g^{\gamma^{2n}}, \tilde{g}_1 = \tilde{g}^{\gamma^1}, \ldots, \tilde{g}_n = \tilde{g}^{\gamma^n}, \tilde{g}_{n+2} = \tilde{g}^{\gamma^{n+2}}, \ldots, \tilde{g}_{2n} = \tilde{g}^{\gamma^{2n}}, z = e(g, \tilde{g})^{\gamma^{n+1}})$.

3. Generate a key pair for the AHO signatures:

$$\begin{aligned} pk_{\mathrm{AHO}} &= (G_r, H_r, G_z, H_z, G, H, A, B) \\ sk_{\mathrm{AHO}} &= (\alpha_a, \alpha_b, \mu_z, \nu_z, \mu, \nu) \end{aligned}$$

4. Generate a CRS for the perfect sound setting of the GS proof: Select $(\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{v}_1, \boldsymbol{v}_2)$ for $\boldsymbol{u}_1 = (u_{11}, u_{12})$, $\boldsymbol{u}_2 = (u_{21}, u_{22})$, $\boldsymbol{v}_1 = (v_{11}, v_{12})$, $\boldsymbol{v}_2 = (v_{21}, v_{22})$, where $u_{11}, u_{12} \in_R \mathbb{G}_1$, $v_{11}, v_{12} \in_R \mathbb{G}_2$ and $\boldsymbol{u}_2 = \boldsymbol{u}_1^{\xi_1}$, $\boldsymbol{v}_2 = \boldsymbol{v}_1^{\xi_2}$ for $\xi_1, \xi_2 \in_R \mathbb{Z}_p^*$.
5. Output the issuer public key ipk=$(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \tilde{g}, pk_{acc},\ pk_{\mathrm{AHO}}, (\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{v}_1, \boldsymbol{v}_2))$, and the issuer secret key isk=sk_{AHO}.

CertObtain. In this protocol, the common inputs are ipk and the user's attribute set U, and the issuer's input is isk. Note that user's attribute set is fixed to a subset of $\{1, \ldots, n\}$, i.e., the small universe.

1. As all possible subsets of set U, the issuer prepares U_k for all $1 \le k \le K$ where K is the total number of the subsets.
2. Using sk_{AHO}, the issuer generates each AHO signatures on $P_k = \prod_{i \in U_k} \tilde{g}_i$ as σ_k for all $1 \le k \le K$ and then send them to the user.
3. The user outputs the obtained signatures $cert = \{(\sigma_k)_{1 \le k \le K}\}$, as the certificates.

ProofGen. The inputs are ipk, U, $cert$ and the monotone formula $\mathcal{M}$. Define $\hat{U} \subseteq U$ is the minimal attribute set selected by the user to satisfy the formula $\mathcal{M}$.

1. Using **AccGen**, run the tag assignment algorithm. For each attribute i in $\mathcal{M}$, a series of tags $\mathcal{S}_i$ is assigned, where the tags are $c_1, \ldots, c_T$. Then, compute the accumulator:

$$acc_{\mathcal{M}} = \prod_{i \in \mathcal{M}_{\mathcal{A}}} g_{n+1-i}^{\sum_{t \in \mathcal{S}_i} c_t}.$$

2. User calculates $P_{\hat{U}} = \prod_{i \in \hat{U}} \tilde{g}_i$.
3. User selects certificate σ_k w.r.t. $P_{\hat{U}}$ s.t. $\hat{U} = U_k$ for some U_k from $cert$.
4. Compute the witness that $\hat{U}$ satisfies $\mathcal{M}$ for $acc_{\mathcal{M}}$:

$$W = \prod_{j \in \hat{U}} \prod_{i \in \mathcal{M}_{\mathcal{A}}, i \neq j} \tilde{g}_{n+1-i+j}^{\sum_{t \in \mathcal{S}(i)} c_t}$$

and sets a public data $u = c_1 + \ldots + c_T$.
5. Compute GS commitments $com_{P_{\hat{U}}}, com_W$ to $P_{\hat{U}}, W$. Then re-randomize the AHO signature σ_k to obtain $\sigma'_k = \{\theta'_1, \ldots, \theta'_7\}$, and compute GS commitments $\{com_{\theta'_i}\}_{i \in \{1,2,5\}}$.
6. Generate the GS proofs $\{\pi_i\}_{i=1}^{3}$ s.t.

$$z^u = e(acc_{\mathcal{M}}, P_{\hat{U}}) \cdot e(g, W)^{-1}, \tag{3.2}$$

$$A \cdot e(\theta'_3, \theta'_4)^{-1} = e(G_z, \theta'_1) \cdot e(G_r, \theta'_2) \cdot e(G, P_{\hat{U}}), \tag{3.3}$$

$$B \cdot e(\theta'_6, \theta'_7)^{-1} = e(H_z, \theta'_1) \cdot e(H_r, \theta'_5) \cdot e(H, P_{\hat{U}}), \tag{3.4}$$

where $z^u, acc_{\mathcal{M}}, g, A, B, \theta'_3, \theta'_4, \theta'_6, \theta'_7, G_z, G_r, G, H_z, H_r, H$ are public data, while $P_{\hat{U}}, W, \theta'_1, \theta'_2, \theta'_5$ are secret data, which are committed.
7. Output $\sigma = (\{\theta'_i\}_{i=3,4,6,7}, com_{P_{\hat{U}}}, com_W, \{com_{\theta'_i}\}_{i=1,2,5}, \{\pi_i\}_{i=1}^{3})$.

The equation (3.2) shows the verification relation of accumulator:

$$\frac{e(acc_{\mathcal{M}}, P_{\hat{U}})}{e(g, W)} = z^u, \tag{3.5}$$

where $P_{\hat{U}} = \prod_{i \in \hat{U}} \tilde{g}_i$. The equations (3.3),(3.4) show the knowledge of the AHO signature of $P_{\hat{U}}$.

Verify. The inputs are *ipk*, the proof σ, and the proved formula $\mathcal{M}$.

1. Using **AccGen**, run the tag assignment algorithm. For each attribute i in $\mathcal{M}$, a series of tags $\mathcal{S}_i$ is assigned, where the tags are $c_1, \ldots, c_T$. Then, compute the accumulator:

$$acc_{\mathcal{M}} = \prod_{i \in \mathcal{M}_{\mathcal{A}}} g_{n+1-i}^{\sum_{t \in \mathcal{S}_i} c_t}.$$

2. Accept σ, if the verification of all GS proofs are successful.

3.4.3 Security

We can prove the following security of our construction.

Theorem 4. *The proposed system satisfies the misauthentication resistance under the security of the AHO signatures and the extended accumulators.*

Proof. To win the misauthentication resistance game, the adversary $\mathcal{A}$ must output a valid proof, when the attributes of corrupted users do not satisfy the predicate $\mathcal{M}^*$. Let $\sigma = (\{\theta'^*_i\}_{i=3,4,6,7}, com^*_{P^*_{\tilde{U}^*}}, com^*_{W^*}, \{com^*_{\theta'^*_i}\}_{i=1,2,5}, \{\pi^*_i\}^3_{i=1})$ be the forged proof. In this proof of theorem, we use notation $\tilde{U}^*$ instead of $\hat{U}$ because $\mathcal{MS}(\tilde{U}^*, \mathcal{M}^*) = 0$. Since the CRS for the perfect soundness setting is prepared, due to the extractability, the GS commitments are extractable. Thus, we can extract $P^*_{\tilde{U}^*}, W^*$ satisfying the equation (3.2) for accumulator verification with $acc^*_{\mathcal{M}^*}$ that is correctly computed from $\mathcal{M}^*$ by the verifier, and the re-randomized AHO signature $\sigma'^*_{\tilde{U}^*} = \{\theta'^*_1, \ldots, \theta'^*_7\}$ for $P^*_{\tilde{U}^*}$ satisfying the equations (3.3),(3.4). We distinguish the following cases.

- **Type 1 forgery.** This is the case that the AHO signature on $P^*_{\tilde{U}^*}$ was never issued to any corrupted user i (i.e., $\mathsf{i} \in CU$).
- **Type 2 forgery.** This is the case that the AHO signature on $P^*_{\tilde{U}^*}$ was issued to a corrupted user i.

Using Type 1 forgery, we can obtain a forger against the AHO signatures, as follows.

Type 1 forgery. The public key pk_{AHO} of AHO signatures is given. Then, choose and compute other parameters in *ipk*, as the real algorithm, and run $\mathcal{A}$ on *ipk*. For the **C-Issuing** query, to the signing oracle, request the AHO signatures on $P_k = \prod_{i \in U_k} \tilde{g}_i$, for all subsets U_k of $U^{(\mathsf{i})}$ requested by $\mathcal{A}$. Respond the AHO signatures as $cert_{\mathsf{i}}$. Finally, $\mathcal{A}$ outputs a predicate $\mathcal{M}^*$, and a proof σ^*. In this case, since the AHO signature $\sigma'^*_{\tilde{U}^*}$ was never issued for $P^*_{\tilde{U}^*}$, this implies the forgery against the AHO signature.

Table 3.1: Asymptotic Efficiency Comparisons.

	ProofGen Cost			**Verify** Cost			Proof size	Certificate size
	EXP	MUL	PAIRING	EXP	MUL	PAIRING		
Previous System [23]	$O(T)$	$O(A \cdot \lvert U \rvert)$	$O(1)$	$O(T)$	$O(A)$	$O(1)$	$O(1)$	$O(1)$
Proposed System	$O(T')$	$O(A' \cdot \lvert \hat{U} \rvert)$	$O(1)$	$O(T')$	$O(A')$	$O(1)$	$O(1)$	$O(2^{\lvert U \rvert})$

T: number of ANDs in CNF formula, T': number of ANDs in monotone formula, A: number of attributes in CNF formula, A': number of attributes in monotone formula, $|U|$: number of user's attributes, $|\hat{U}|$: number of attributes in minimum attribute set.

Using Type 2 forgery, we can obtain an adversary against the extended accumulator, as follows.

Type 2 forgery. The public parameters of the extended accumulator are given. Then, choose and compute other parameters in ipk, as the real algorithm, and run $\mathcal{A}$ on ipk. In the run, each **C-Issuing** query is responded as in the real algorithm. Finally, $\mathcal{A}$ outputs a predicate $\mathcal{M}^*$, and a proof σ^*. In this case, the AHO signature on $\tilde{U}^*$ was correctly issued to some corrupted user i. Thus $P^*_{\tilde{U}^*} = \prod_{i \in \tilde{U}^*} \tilde{g}_i$ for a subset $\tilde{U}^*$ of $U^{(\mathsf{i})}$, and $\tilde{U}^*$ does not satisfy $\mathcal{M}^*$. Note that $acc^*_{\mathcal{M}^*} = \prod_{i \in \mathcal{M}^*_{\mathcal{A}}} g_{n+1-i}^{\sum_{t \in \mathcal{S}_i} c_t}$ is correctly computed by the verifier. Therefore, we can forge $\tilde{U}^*, \mathcal{M}^*$, and W^*, where $P^*_{\tilde{U}^*}$ and $acc^*_{\mathcal{M}^*}$ are correct, **AccVerify** accepts $\tilde{U}^*, W^*, acc^*_{\mathcal{M}}$, but $\mathcal{MS}(\tilde{U}^*, \mathcal{M}^*) = 0$.

□

Theorem 5. *The proposed system satisfies the anonymity under the SXDH assumption.*

Proof. Consider the sequence of games, as follows.

Game 1. This is the anonymity game for the proposed system. The challenger generates ipk, isk using **IssuerKeyGen** algorithm, where the CRS is prepared for the perfect soundness setting. The challenger runs the adversary $\mathcal{A}$ with ipk, isk. For the **Proving** query and the challenge query, the challenger responds using ipk and $cert_{\mathsf{i}}$ in the response of the **H-Issuing** query.

Game 2. In **IssuerKeyGen** algorithm, the challenger generates the CRS for the perfect WI setting. Namely, choose $(\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{v}_1, \boldsymbol{v}_2)$ for $\boldsymbol{u}_1 = (u_{11}, u_{12})$, $\boldsymbol{u}_2 = (u_{21}, u_{22})$, $\boldsymbol{v}_1 = (v_{11}, v_{12})$, $\boldsymbol{v}_2 = (v_{21}, v_{22})$, where $u_{11}, u_{12} \in_R \mathbb{G}_1$, $v_{11}, v_{12} \in_R \mathbb{G}_2$ and $\boldsymbol{u}_2 = \boldsymbol{u}_1^{\xi_1}/(1, \tilde{g})$, $\boldsymbol{v}_2 = \boldsymbol{v}_1^{\xi_2}/(1, \tilde{g})$ for $\xi_1, \xi_2 \in_R \mathbb{Z}_p^*$. The others are the same as Game 1.

Let S_1, S_2 denote the events that $\phi' = \phi$ in Game 1, 2, respectively. In Game 2, the proof of responded in the challenge consists of the GS commitments that are perfectly hiding in the WI setting, the GS proofs that reveal no information about the underlying witness due to the perfect **WI**, and the randomized AHO signatures $\{\theta'_i\}_{i=3,4,6,7}$ that are information-theoretically independent of the signed messages and the remaining AHO signatures. Thus, we have $\Pr[S_2] = 1/2$. On the other hand, $|\Pr[S_1] - \Pr[S_2]|$ is negligible due to the CRS indistinguishability under the SXDH assumption. Therefore, the advantage of $\mathcal{A}$, i.e., $|\Pr[S_1] - 1/2|$, is negligible, which means that the proposed system is anonymous. □

3.5 Comparisons and Efficiency Improvement

As mentioned in Introduction, since there are the previous systems of [15, 1, 21, 23] with constant-size attribute proofs, we briefly discuss about the comparisons. The previous systems [15, 1] support only simple AND or OR relations on attributes, and thus the proved formulas are less expressive. The previous system [21] supports the AND/OR relation as an inner product on two vectors. The proof generation requires $O(1)$ pairing but $O(n^2)$ exponentiations, where n is the size of vectors (The verifying costs are $O(1)$ pairing and $O(n)$ exponentiations). As shown in [17], using the inner product, CNF and DNF formulas on attributes are verified via polynomial evaluations. In the attribute proof, the vector size depends on the number of OR relations in the proved formula. This is why the proof generation suffers from the heavy exponentiation costs in cases of formulas with lots of OR relations, which is our targets, such as the example of the alcohol related website. Therefore, in the folowing detailed comparisons, we concentrate on the remaining system [23].

3.5.1 Efficiency Comparisons

We compare the efficiency of our proposed system to the previous system [23].

Firstly, we compare the asymptotic efficiency of the proof (**ProofGen**'s output σ) size, certificate size, and the computation costs of the more frequently executed authentication protocol that consists of **ProofGen** and **Verify** algorithms. In both systems, **ProofGen** mainly consists of computations of $acc_{\mathcal{M}}$ and W and GS proof generation. **Verify** consists of the computation of $acc_{\mathcal{M}}$ and GS proof verification.

Here, we review the computations of W from the previous system and the proposed system. In the previous system,

$$W = \prod_{j \in U} \prod_{1 \le t \le T} \left(\prod_{j \in V_t}^{i \ne j} \tilde{g}_{n+1-i+j} \right)^{c_t},$$

where c_t are similar tags, and V_t includes the attribute literals of the t-th clause in the CNF formula. In this computation, by arranging the calculation order, the cost of the exponentiations of c_t can be reduced to T, which is the number of ANDs. On the other hand, the number of multiplications is $A \cdot |U|$, where A is the number of attribute literals in CNF formula, i.e., $A = \sum_{1 \le t \le T} |V_t|$. Similarly, we can observe that the exponentiation and multiplication costs to compute the accumulator are about T and A, respectively.

In the proposed system,

$$W = \prod_{j \in \hat{U}} \prod_{i \in \mathcal{M}_{\mathcal{A}}, i \ne j} \tilde{g}_{n+1-i+j}^{\sum_{t \in S_i} c_t}.$$

Also, we can arrange the calculation order to

$$W = \prod_{1 \le t \le T'} (\prod_{j \in \hat{U}} (\prod_{i \in S_{\mathcal{M}_{\mathcal{A}}}{}^{-1}(t), i \ne j} \tilde{g}_{n+1-i+j}))^{c_t},$$

where T' is the number of AND in the monotone formula, and $S_{\mathcal{M}_{\mathcal{A}}}{}^{-1}(t)$ is the set of i such that attribute i is assigned to tag c_t. The number of exponentiations of c_t is approximately T'. The number of multiplications is $A' \cdot |\hat{U}|$, where A' denotes the number of attribute literals in $\mathcal{M}$, i.e., $A' = |\mathcal{M}_{\mathcal{A}}|$, and it is at most $A' \cdot |U|$, due to $|\hat{U}| \le |U|$. Similarly, we can observe that the exponentiation and multiplication costs to compute the accumulator are about T' and A', respectively.

Table 3.1 summarizes the asymptotic efficiency comparisons, where EXP, MUL, and PAIRING mean the costs of exponentiations, multiplications, and pairings, respectively. Note that the computation and size of GS proofs used in **ProofGen** and **Verify** does not depend on the parameters $T, T', A, A', |U|, |\hat{U}|$. This is similar to [23] where the number of GS proofs is reduced and thus the computational cost is reduced from the previous system to the proposed system. Thus, the pairing cost is $O(1)$ in both systems, and the size of proof σ outputted in **ProofGen** is also $O(1)$ in the both systems. On the other hand, although the certificate size is $O(1)$ in [23], that of the proposed system is $O(2^{|U|})$, since $2^{|U|}$ AHO signatures are issued.

As shown in Table 3.1, the computation costs in both systems are comparable, if the parameters A and T for the CNF formula in the previous system are the *same* as A' and T' for the monotone formula in the proposed system. However, note that, since the monotone formula is more expressive, the representaion of the monotone formula for a proved formula may be shorter than the CNF formula, i.e., A' and T' may be smaller than A and T. Any monotone formula can be converted to a CNF formula. However, the conversion may cause A and T for the converted CNF formula to grow larger than A' and T' for the original monotone formula. For example, a monotone formula $(a_{11} \wedge a_{12}) \vee (a_{21} \wedge a_{22}) \vee \cdots (a_{k1} \wedge a_{k2})$ with $O(k)$ literals can be converted to a CNF formula $(a_{11} \vee a_{21} \vee \cdots \vee a_{k1}) \wedge (a_{11} \vee a_{22} \vee \cdots \vee a_{k1}) \wedge \cdots (a_{12} \vee a_{22} \vee \cdots \vee a_{k2})$ with $O(2^k)$ literals by using the distribution property. Thus, in the cases that proved formulas require longer sizes in the representation of the CNF formula than the monotone formula, such as this example, the proposed system has more efficient computation costs, since $O(T)$ exponentiations (resp., $O(A \cdot |U|)$ multiplications) need more computations than $O(T')$ exponentiations (resp., $O(A' \cdot |\hat{U}|)$ multiplications) due to $A' < A$ and $T' < T$. This is the main advantage of our system.

Next, we show the concrete efficiency using an example of authentication for accessing alcohol related websites using nationality and birth date, which is the situation of lots of OR relations in the proved formula, as targeted in Section 3.1. In the formula, four categories of attributes are used; the nationality, the birth-year, the birth-month and the birth-day. The example of the monotone formula is as follows:
$F_1 = (\mathit{Australia} \vee \ldots) \wedge (1915 \vee \ldots \vee (1997 \wedge (\mathit{Jan} \vee \ldots \vee (\mathit{Sept} \wedge (1^{st} \vee \ldots \vee 5^{th}))))).$
The CNF type of formula is as follows:
$F_2 = (\mathit{Australia} \vee \ldots) \wedge (1915, \mathit{Jan}.1^{st} \vee \cdots \vee 1997, \mathit{Sept}.5^{th}),$
where each birthday is encoded to one attribute value such as "1915, $\mathit{Jan}.1^{st}$".

In the previous system [23] for CNF formulas, as described in Section **??**, F_2 has $A = 30,299$ attribute literals, and thus the multiplication costs with $O(A \cdot |U|)$ complexity becomes very large. On the other hand, in the proposed system, F_1 needs $A = 198$ attribute literals, and the multiplication cost with $O(A' \cdot |\hat{U}|)$ complexity is greatly reduced. In F_1, the number of ANDs is small (concretely, 3), and thus the exponentiation cost is small. The concrete experiment of implementation is shown in Section 3.6.3.

As indicated by the above example, the proposed system for monotone formulas is more advantageous over [23] in cases of numerical range proofs. Another application example of range proofs is to verify the expiry date in a privacy-enhancing way. For example, in an online video streaming service, a user can show his access privilege to the service by showing his subscription and the expiry-date. To conceal the expiry-date for anonymity, the formula of expiration check can be expressed as the range from today to the maximum of subscription

Table 3.2: Comparison of computation times.

	Proposed System (monotone formula)	Previous System[23] (CNF formula)
Prover Time $[ms]$	63.04	969.11
Verifier Time $[ms]$	132.57	376.97

Table 3.3: Comparison of $acc_{\mathcal{M}}$ and W times.

	Proposed system	Previous system
$acc_{\mathcal{M}}$ Time $[ms]$	3.24	159.19
W Time $[ms]$	13.89	748.18

time, where the user proves that his expiry-date is in the range. The monotone formula is expressed efficiently as well as the above F_1, and thus the range proof is efficiently executed, compared to [23].

3.5.2 Comparison of Restrictions

A formula that can be used in the proposed system is in monotone formulas, while that in the previous system [23] is in CNF which is more restricted class. However, due to the underlying accumulators, restrictions on the usable formulas and user's attributes exist in both systems, as follows. Firstly, the accumulator works well, only when $T < \log_2 p / \log_2(\eta+1)$. This is because we assume $(\eta+1)c_T < p$ in **AccGen**, which implies $p > (\eta+1)c_T = (\eta+1)^T$, and thus $\log_2 p > \log_2(\eta+1)^T$. Since $\log_2 p > \log_2(\eta+1)^T = T\log_2(\eta+1)$, we have the restriction $T < \log_2 p / \log_2(\eta+1)$, i.e., the number T of ANDs in the formula must be less than $\log_2 p / \log_2(\eta+1)$, where η is th upper bound of $|U|$. For example, in the 128 bit security with 254-bit group order and $\eta = 50$, we can set $T \approx 40$ in maximum. Since one person in general owns at most 40 attributes for user's attribute authentications, we consider that this restriction is not too strong. This restriction also exists in the previous scheme [23], due to the similar construction of the accumulator and similar assumption for c_T.

Secondly, the proposed system is of the *small universe*, i.e., the set of all attributes is fixed in advance by some authority and users have to select their attributes from the fixed set, since the proposed system needs that the attribute set is correspondent to $\{1, \ldots, n\}$. The *large universe*, i.e., the attribute set is not fixed, is general, and thus the small universe is a restriction. This restriction also exists in the previous scheme [23], since the construction of the accumulator is similar.

Thirdly, the proposed system has a restriction that the upper bound of $|U|$ of every user, η, is fixed in advance by some authority. On the other hand, the previous system has other restrictions: For the t-th OR clause in every proved CNF formula and every user's attribute set U, the maximum number of $|V_t \cap U|$ has to be fixed in advance. Also, the maximum number of OR clauses in a CNF formula has to be fixed.

Fourthly, in the proposed system, we assume that the attribute indices in $\mathcal{M}$ are all different, due to the underlying accumulator, even when $\mathcal{M}$ inlcudes the same attributes twice or more. The previous system [23] does not need the assumption. As mentioned in Section 3.2.5, we can cope with this by assigning the same attributes in $\mathcal{M}$ to different attribute indices, but the number of indices for the same attribute has to be fixed in advance.

3.5.3 Reducing Certificate Size

As mentioned in Section 3.4.1, we consider that the user is issued signatures $cert = (\sigma_k)_{1\leq k\leq K}$ for all subsets $U_k (1 \leq k \leq K)$ for the set of user's attributes, U. Thus, the certificate size becomes large, since K is increased exponentially to the number of user's attributes $|U|$. To reduce the number of the certificates, we can show a simple improvement idea. The idea is to separate set U into two subsets of U_1 and U_2. We consider $|U_1| \simeq |U_2| \simeq |U|/2$. The issuer generates signatures of all subsets of U_1 and signatures of all subsets of U_2 independently. The user obtains $cert_1 = (\sigma_k)_{1\leq k\leq K_1}$ and $cert_2 = (\sigma_k)_{1\leq k\leq K_2}$, where K_1 and K_2 are the numbers of subsets in set U_1 and U_2, respectively. In the attribute proof protocol, the user proves the knowledge of the signatures of both $\hat{U}_1$ and $\hat{U}_2$ using GS Proof for $P_{\hat{U}_1} = \prod_{i\in\hat{U}_1} \tilde{g}_i$ and $P_{\hat{U}_2} = \prod_{i\in\hat{U}_2} \tilde{g}_i$ respectively, where $\hat{U}_1 \subseteq U_1$ and $\hat{U}_2 \subseteq U_2$. Hence, the accumulator verification equation is modified as follows:

$$\frac{e(acc_{\mathcal{M}}, P_{\hat{U}_1})e(acc_{\mathcal{M}}, P_{\hat{U}_2})}{e(g, W)} = z^u.$$

By this equation, the user proves that his/her attributes from subset $\hat{U}_1$ and $\hat{U}_2$ satisfy the monotone formula $\mathcal{M}$. Compared to the proposed system in section 5.2, this modification idea reduces the certificate size from $K = 2^{|U|}$ into approximately $\sqrt{K}$.

3.6 Implementation and Experiment

To show the practicality of our system, we implemented the system and measured the processing time. In this section, we show the experimental results.

3.6.1 Utilized Pairing Library

At the coming of 128-bit security, the asymmetric pairing e s.t. $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is faster than the symmetric one [24]. Thus, in the implementation, we adopt the asymmetric type, and we utilize the fast pairing library called "Cross-twisted χ-based Ate (Xt-Xate) pairing" [20] with 254-bit group order and the embedding degree is 12. The security level is equivalent to the 128-bit AES. The library is based on the GMP library and implemented by C language due to the pursuit of the fastness.

3.6.2 Instantiation of GS Proof

For the implementation, we need to instantiate the GS proof [16] concretely. Based on the utilized pairing and cryptographic assumption, there are three types of instantiations. Since we utilize asymmetric pairing from the viewpoint of efficiency, we adopt the GS proof based on the SXDH assumption for the asymmetric pairing.

Table 3.4: Environments of implementation and experiments.

CPU	Intel Core i5-4460(3.20GHz)
Main memory	7.8 GB
OS	Ubuntu 14.04 LTS
Multiple Precision Arithmetic Library	GMP-6.0.0
Pairing Library	ELiPS [20]
C compiler	GCC-4.8.4

3.6.3 Experiment and Evaluation

To confirm the efficiency consideration of the comparison between the proposed system for monotone formulas and the previous system [23] for CNF formulas in Section 3.5, we measured the processing times of the authentication protocol for the prover (**ProofGen**) and verifier (**Verify**). The environments of the implementation and experiments are shown in Table 5.1.

Table 3.2 shows the processing times. The used formulas are F_1 and F_2 in Section 3.5, where the monotone formula F_1 includes 198 literals, and the CNF formula F_2 includes 30,299 literals. From this table, we can confirm that both prover and verifier times are reduced, but the prover time is greatly reduced. To explore the reason, we measured the computations of $acc_{\mathcal{M}}$ and W that depend on the formula size, as in Table 3.3. From this table, we can confirm that the reduction of these computation times (especially W time) influences the reduction of prover and verifier times.

3.7 Conclusions

In this chapter, we proposed an extended accumulator to prove monotone formulas on attributes, and apply it to the anonymous credential system in order to obtain more efficiency in the proofs generation. The monotone formula is more expressive and compact than the CNF formulas, and thus the proposed system can reduce the proof generation time, compared to the previous system for CNF formulas.

Then, we also implemented the scheme on an example of age authentication to show the practicality. The result shows that the proposed scheme reduced both sign and verify times of the previous scheme. Although the size of user's certificates is $O(2^{|U|})$ due to the restriction of the newly proposed accumulator for monotone formulas, a simple modification idea enables the scheme to reduce the size from $O(2^{|U|})$ to $O(\sqrt{2^{|U|}})$.

Chapter 4

Revocable Group Signatures with Compact Revocation List Using Vector Commitments

4.1 Introduction

In a group signature scheme [8], a group member is allowed to sign a message anonymously on behalf of the group. There are two types of authorities: A *group manager* (GM) who adds users into the group, and an *opener* who can identify the signer from the signature when necessary. One important function in the group signature scheme is *revocation*, where the user's privilege to sign a message is removed. It is a critical issue, which has been broadly studied.

Hereafter, let N be the total number of group members, and R be the number of revoked members. On the early study [10], the signature size, and also the singing and verification costs are $O(R)$. Then, the accumulator-based scheme has been proposed to achieve the constant-size signature with the constant verification costs [13]. However, each member has to update a secret key (a witness for the accumulator) using the revocation data, which implies that singing costs is $O(R)$ in the worst case. On the other hand, verifier-local revocation (VLR) has been proposed [5], [29]. In this approach, the revocation data are sent to only verifiers, and thus the signer's load is minimum. However, the verification includes the revocation check with $O(R)$ complexity. Then, a revocable scheme with constant singing and verification costs have been proposed [27]. The demerit of this scheme is the long public key size. The basic scheme needs $O(N)$ size, and the extended one needs $O(\sqrt{N})$ in exchange for the extra signing cost.

Recently, Libert et al. proposed a scalable scheme [4] based on the broadcast encryption framework by Naor et al. [9], where $O(1)$ signature size, $O(1)$ signing/verification costs, $O(1)$ membership certificate size, and $O(\log N)$ public key size are achieved. However, the scheme still needs an improvement on the revocation list (RL) size. In the scheme, the RL contains signatures for all subsets of authorized users, which are formed by a subset difference (SD) method. In the worst case, the number of signatures amounts to $2R-1$. As the signature, an AHO signature with 7 group elements is used. Thus, in case of 128-bit security, the RL size is $900R$ bytes or more. Since the signer needs to fetch the RL for every revocation epoch, the large size will cause delay in mobile environment.

There are studies on reducing the RL size. Nakanishi et al. proposed a scheme [28] with compact RL using an accumulator. In this scheme, since GM accumulates T SD subsets and signs the accumulated value, the number of signatures is reduced by $1/T$ and the RL size is $O(R/T)$. However, the public key size and membership certificate size are increased, when T is increased. The other scheme is proposed by Attrapadung et al. [22], where the RL size is constant by adopting identity based revocation (IBR) method. However, as the trade-off, the membership certificate size and signing cost are $O(R)$ in the worst case.

In this chapter, we propose a revocable group signature scheme with a compact RL. Similarly to [28], we partition the subsets into a number of blocks and compress it using a vector commitment [2], where each commitment can be opened w.r.t. individual coordinates in a space-efficient manner. Since the compression is simpler than the accumulator in [28], we can reduce the RL size to $O(R/T)$, while maintaining the membership certificate size as $O(1)$. The public key size is $O(T + \log N)$ due to the compressions for RL and the user's membership certificate. In section 4.5, we compare the RL size between the proposed scheme and [28] as shown in Table 4.2. Then, we also show the comparison of RL size between the proposed scheme and [4] when we set $T = \log N$. In the discussion, we also clarify that the $O(T)$ signing time is a trade-off to the reduction of RL size.

4.2 Syntax and Security of Revocable Group Signatures

As in [4], we define the revocable group signatures.

4.2.1 Syntax

The algorithms and protocol of the revocable group signature scheme are as follows.

Setup: Given a security parameter $\lambda \in \mathbb{N}$, a maximal number of a group members $N \in \mathbb{N}$ and a partitioning parameter $T \in \mathbb{N}$, this algorithm generates a group public key $\mathcal{Y}$, the group manager's (GM) private key $\mathcal{S}_{\texttt{GM}}$ and the opener's private key $\mathcal{S}_{\texttt{OA}}$. This algorithm initializes a public state St comprising a set data structure $St_{users} = \phi$ and a string data structure $St_{trans} = \epsilon$.

Join: This interactive protocol is between GM and prospective group's member u. The interactive Turing machines are denoted as $\texttt{J}_{\texttt{GM}}$ and $\texttt{J}_{\texttt{user}}$, respectively. The common input is $\mathcal{Y}$. GM's additional inputs are St and $\mathcal{S}_{\texttt{GM}}$. As a result, the member obtains a membership secret $\texttt{sec}_u$ and a membership certificate $\texttt{cert}_u$. GM updates St in the database by $St_{user} = St_{user} \cup \{u\}$ and $St_{trans} = St_{trans} || < u, \texttt{transcript}_u >$.

Revoke: This algorithm is run by GM. The inputs are $\mathcal{Y}$, $\mathcal{S}_{\texttt{GM}}$, epoch t, and the set of the revoked users, $\mathcal{R}_t$. This algorithm allows GM to generate an update revocation list RL_t for the new revocation epoch t.

Sign: This algorithm is run by u. Given $\mathcal{Y}, t, \mathrm{RL}_t$, $\texttt{cert}_u, \texttt{sec}_u$, and a message M, this algorithm outputs $\perp$ if $u \in \mathcal{R}_t$, or a signature σ otherwise.

Verify: This algorithm is run by a verifier. Given $\mathcal{Y}, t$, RL_t, σ, M, this deterministic algorithm outputs 1 if the signature is valid and not revoked in RL_t, or 0 otherwise.

Open: This algorithm is run by an opener. Given $\mathcal{Y}$, $t, \mathrm{RL}_t, \sigma, M, St, \mathcal{S}_{\mathtt{OA}}$, it outputs $u \in St_{users} \cup \{\perp\}$, which is the identity u as a group member or an opening failure.

4.2.2 Security

There are three security requirements in the revocable group signature scheme. The first one is *security against misidentification attacks*. This requirement means that an adversary cannot compute a group signature where **Open** identifies outside of the corrupted non-revoked members. The second one is *security against framing attacks*, which means that an honest member is not traced for signatures that the member did not issue, even if everyone (including GM) except the member colludes. The third one is *anonymity*, which implies the anonymity and unlinkability of signatures.

The formal definitions are as follows. The definition of the correctness is omitted due to the page limitation. The security model is formalized by experiments between the adversary and a stateful interface $\mathcal{I}$. Let $\mathsf{state}_{\mathcal{I}}$ be a data structure which is employed by $\mathcal{I}$, and $\mathsf{state}_{\mathcal{I}}$ is initialized as $(St,\, \mathcal{Y},\, \mathcal{S}_{\mathtt{GM}},\, \mathcal{S}_{\mathtt{OA}}) \leftarrow \mathbf{Setup}(\lambda,\, N,\, T)$.

The adversary can access to the following oracles.

$Q_{\mathcal{Y}}$, $Q_{\mathcal{S}_{\mathtt{GM}}}$, $Q_{\mathcal{S}_{\mathtt{O}}}$: The interface looks up $\mathsf{state}_{\mathcal{I}}$ and responds $\mathcal{Y}$, $\mathcal{S}_{\mathtt{GM}}$ and $\mathcal{S}_{\mathtt{OA}}$, respectively.

Q_{a-join}: This is the join of a corrupted user. The interface simulates $\mathtt{J}_{\mathtt{GM}}$, and interacts with the adversary executing $\mathtt{J}_{\mathtt{user}}$. In addition to St_{user}, u is added to the set U^a, which includes the identities of corrupted users.

Q_{b-join}: This is the join of an honest user. The interface simulates $\mathtt{J}_{\mathtt{user}}$, while the adversary plays the role of $\mathtt{J}_{\mathtt{GM}}$. In addition to St_{user}, u is added to the set U^b, which includes the identities of honest users. The outputted $\mathtt{cert}_u$, $\mathtt{sec}_u$ are appended to a private area of $\mathsf{state}_{\mathcal{I}}$.

Q_{sig}: To the query on message M and index u , the interface checks if $(\mathtt{cert}_u,\, \mathtt{sec}_u)$ exists in the private area of $\mathsf{state}_{\mathcal{I}}$ such that $u \notin \mathcal{R}_t$ for the current epoch t. If no entry in the area or $u \notin U^b$, the interface returns $\perp$. Otherwise, it returns the signature σ of u at the current epoch t, and stores $(u,\, t,\, M,\, \sigma)$ in the database history Sigs.

Q_{open}: To the query on a signature-message pair $(\sigma,\, M)$ at epoch t, the interface executes **Open** on the current state St.

Q_{read}, Q_{write}: By these queries, the adversary can read or write the contents of $\mathsf{state}_{\mathcal{I}}$. Q_{read} returns the whole $\mathsf{state}_{\mathcal{I}}$ excluding the public key and secret keys and the private area used in Q_{b-join}. Q_{write} allows the adversary to modify $\mathsf{state}_{\mathcal{I}}$ as long as it does not remove or corrupt elements of St_{users}, St_{trans} (the adversary can add a dummy user).

Q_{revoke}: To the query on $u \in St_{users}$, the interface checks if u is included in U^a or U^b depending on the considered security notion and if St_{trans} contains $\langle u, \mathtt{transcript}_u \rangle$ such that $u \notin \mathcal{R}_t$ for the current epoch t. If not, it returns $\perp$. Otherwise, it increments t, add u to $\mathcal{R}_t$, generate the revocation list RL_t which is available to the adversary.

Here, we formally define only the *security against misidentification attacks*, since our modification to the underlying scheme [4] does not influence the other requirements.

Definition 6: A revocable group signature scheme is secure against misidentification attacks if, for any PPT adversary $\mathcal{A}$ involved in experiments $\mathbf{Exp}_{\mathcal{A}}^{\text{mis-id}}(\lambda)$, the probability $\mathbf{Adv}_{\mathcal{A}}^{\text{mis-id}}(\lambda) = \Pr[\mathbf{Exp}_{\mathcal{A}}^{\text{mis-id}}(\lambda) = 1]$ is negligible for λ.

```
Experiment $\mathbf{Exp}_{\mathcal{A}}^{\text{mis-id}}(\lambda)$
    $\mathsf{state}_{\mathcal{I}} = (St, \mathcal{Y}, \mathcal{S}_{\mathtt{GM}}, \mathcal{S}_{\mathtt{OA}}) \leftarrow \mathbf{Setup}(\lambda, N, T)$;
    $(M^\star, \sigma^\star) \leftarrow \mathcal{A}(Q_{\mathcal{Y}}, Q_{a-join}, Q_{revoke}, Q_{read}, Q_{\mathcal{S}_{\mathtt{O}}})$;
    If $\mathbf{Verify}(\mathcal{Y}, t^\star, \mathrm{RL}_{t^\star}, \sigma^\star, M^\star)$=0 return 0;
    $u \leftarrow \mathbf{Open}(\mathcal{Y}, t^\star, \mathrm{RL}_{t^\star}, \sigma^\star, M^\star, St, \mathcal{S}_{\mathtt{OA}})$;
    If $(u \notin U^a \setminus \mathcal{R}_{t^\star})$ return 1;
Return 0;
```

In this experiment, $t^\star$ denotes the current revocation epoch when $\mathcal{A}$ outputs $(M^\star, \sigma^\star)$.

4.3 Previous Scheme

In this section, we explain about the revocable group signature proposed by Libert et al. [4], and discuss about the remaining problem. In [4], the approach of the Subset Difference (SD) method is used as in Fig. 1. In a binary tree, group members are assigned to the leaves. Each node v is indexed by an identifier $\mathtt{ID}(v)$ of an integer. When a user becomes a member of the group, the group manager (GM) certifies him with a signature that contains the node IDs of the path from the root to the user's leaf, $(I_1, \ldots, I_\ell)$, where ℓ shows the level of the tree.

In the SD method, non-revoked users are divided into disjoint subsets (subtrees), where a subset S_i is defined by primary node $\mathtt{P}_i$, which is the root node of the subtree, and the secondary node $\mathtt{S}_i$ that is a descendant of $\mathtt{P}_i$. The subset S_i consists of the leaves of the subtree rooted by $\mathtt{P}_i$ except leaves of the subtree rooted by $\mathtt{S}_i$. The levels of $\mathtt{P}_i$ and $\mathtt{S}_i$ are denoted as ϕ_i and ψ_i, respectively.

In the example of Fig.1, the user of the leaf node with $\mathtt{ID}(v)$=8 is distinguished by the path $(I_1, I_2, I_3, I_4) = (1, 2, 4, 8)$. It is included in subset S_1 rooted by $\mathtt{P}_1$ but not by $\mathtt{S}_1$.

For every revocation, GM renews a revocation list RL that contains signed dataset $R_i = (\phi_i, \psi_i, \mathtt{ID}(\mathtt{P}_i), \mathtt{ID}(\mathtt{S}_i))$ of all subset entries $i \in \{1, \ldots, m\}$. For signing a group signature, the user retrieves the revocation data for subset S_i that contains his leaf, and proves that $\mathtt{ID}(\mathtt{P}_i) = I_{\phi_i}$ and $\mathtt{ID}(\mathtt{S}_i) \neq I_{\psi_i}$ to prove the non-revocation. This relation means that his leaf v is connected with $\mathtt{P}_i$ on level ϕ_i and not connected to $\mathtt{S}_i$ on level ψ_i, which means that v is in the sub-tree rooted by $\mathtt{P}_i$ but not in that by $\mathtt{S}_i$. The proof is done in the Zero-Knowledge Proof fashion for the anonymity.

The problem in this scheme is the size of RL. We have $m \leq 2R - 1$ [4]. Each entry is signed by GM using an AHO signature. Furthermore, an AHO signature contains 7 bilinear group elements. For $R = 10,000$, the size of signatures could be about 8MB in 128 bit security. The signer needs to fetch the large RL for every revocation epoch, which will cause delay in mobile environment.

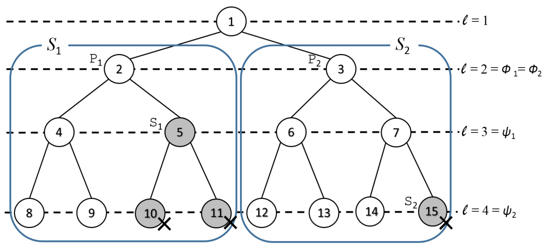

Figure 4.1: An example of SD method.

4.4 Proposed Scheme

4.4.1 Construction Idea

The proposed scheme is extended from the previous scheme [4]. In the previous scheme, GM signs each RL entry of $(\phi_i, \psi_i, \mathtt{ID}(\mathtt{P}_i), \mathtt{ID}(\mathtt{S}_i))$. Meanwhile, in the proposed scheme, multiple elements of each type in the entry are compressed using a vector commitment [2] and are signed. For example, $\mathtt{ID}(\mathtt{P}_1), \ldots, \mathtt{ID}(\mathtt{P}_m)$ are divided to blocks with T elements as $(\mathtt{ID}(\mathtt{P}_1), \ldots, \mathtt{ID}(\mathtt{P}_T)), (\mathtt{ID}(\mathtt{P}_{T+1}), \ldots, \mathtt{ID}(\mathtt{P}_{2T})), \ldots$, and each block of elements is accumulated to the vector commitment. By this accumulation, the number of signed entries in the RL is reduced to m/T.

In [4], the signer only retrieves $(\phi_i, \psi_i, \mathtt{ID}(\mathtt{P}_i), \mathtt{ID}(\mathtt{S}_i))$ of the subtree that contains his leaf v. In the proposed scheme, since we adopt the vector commitment, the signer must show the correct opening of vector commitments. The verification needs the correctness of g_j indicating the opened j-th component. Thus all g_j are signed by GM, and the signer proves the correctness of g_j by the GS proof of the signature of g_j. In addition, we need to prove that g_ϕ and g_ψ are compatible to g_1^ϕ and g_1^ψ in the vector commitment respectively, because we need to map g_1^ϕ and g_1^ψ to their public parameter as g_ϕ and g_ψ. Thus, GM prepares signatures of (g_1^ϕ, g_ϕ) and (g_1^ψ, g_ψ), and the signer proves the mapping, using the GS proofs of the signatures.

4.4.2 Construction

Setup(λ, N, T)**:** Given a security parameter $\lambda \in \mathbb{N}$ and the allowed number of users $N{=}2^{\ell-1}$ for integer ℓ, and the number of elements for partitioning T, do the following, where step 5 and 6 are added to the previous scheme [4].

1. Select bilinear group $(\mathbb{G}, \mathbb{G}_T)$ of prime order $p{>}2^\lambda$, with a generator $g \xleftarrow{R} \mathbb{G}$.

2. Define $\delta_0{=}1$, $\delta_1{=}2$, $\delta_2{=}2$, $\delta_3{=}5$. Generate four key pairs $(sk_{\text{AHO}}^{(d)}, pk_{\text{AHO}}^{(d)})$, $d \in \{0, 1, 2, 3\}$ for the AHO signature in order to sign messages of $\{\delta_d\}_{d=0}^3$ group elements, respectively.

3. Generate a public parameter $pk_{\mathtt{vc}}{=}(g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}) \in \mathbb{G}^{2n-1}$ for n-dimension vector commitment, where $n{=}\mathbf{max}(\ell, T)$.

4. Generate a CRS for the GS NIWI proof: Select $\boldsymbol{f}{=}(\boldsymbol{f}_1, \boldsymbol{f}_2, \boldsymbol{f}_3) \in \mathbb{G}^3$, and $\tilde{\boldsymbol{f}} \in \mathbb{G}$, where $\boldsymbol{f}_1{=}(f_1, 1, g)$, $\boldsymbol{f}_2{=}(1, f_2, g)$, $\boldsymbol{f}_3{=}\boldsymbol{f}_1^{\varepsilon_1} \cdot \boldsymbol{f}_2^{\varepsilon_2}$ with $f_1{=}g^{\beta_1}$, $f_2{=}g^{\beta_2} \xleftarrow{R} \mathbb{G}$ and $\beta_1, \beta_2, \varepsilon_1, \varepsilon_2 \xleftarrow{R} \mathbb{Z}_p^*$. Set $\tilde{\boldsymbol{f}}{=}\boldsymbol{f}_3 \cdot (1, 1, g)$.

5. Using $sk^{(0)}_{\text{AHO}}$, generate an AHO signature on message g_j as $\sigma_{g_j} = (\theta_{j,1}, \ldots, \theta_{j,7})$ for all $1 \le j \le T$.

6. Using $sk^{(1)}_{\text{AHO}}$, generate an AHO signature on the pair of (g_τ, g_1^τ) as $\tilde{\sigma}_\tau = (\tilde{\theta}_{\tau,1}, \ldots, \tilde{\theta}_{\tau,7})$ for all $1 \le \tau \le \ell$.

7. Choose $(U, V) \overset{R}{\leftarrow} \mathbb{G}^2$ that, together with generators $f_1, f_2, g \in \mathbb{G}$, will form a public encryption key.

8. Select a strongly unforgeable one-time signature $\Sigma = (\mathcal{G}, \mathcal{S}, \mathcal{V})$.

9. Set $\mathcal{S}_{\mathtt{GM}} := (\{sk^{(d)}_{\text{AHO}}\}_{d \in \{0,1,2,3\}})$, $\mathcal{S}_{\mathtt{OA}} := (\beta_1, \beta_2)$ as GM's and opener's private keys, respectively, and the group public key as

$$\mathcal{Y} := \Big(\; g, N, \ell, T, \{pk^{(d)}_{\text{AHO}}\}_{d=0}^{3}, pk_{\mathtt{vc}}, \boldsymbol{f}, \tilde{\boldsymbol{f}}, (U, V), \Sigma, \{\sigma_{g_i}\}_{i=1}^{T}, \{\sigma_k\}_{k=1}^{\ell} \; \Big).$$

Join: GM and a joining user u run the following interactive protocol $\mathtt{J}_{\mathtt{user}}(\lambda, \mathcal{Y})$, $\mathtt{J}_{\mathtt{GM}}(\lambda, St, \mathcal{Y}, \mathcal{S}_{\mathtt{GM}})$, which is the same as [4].

1. $\mathtt{J}_{\mathtt{user}}$ chooses $x \overset{R}{\leftarrow} \mathbb{Z}_p$, computes $X = g^x$ and send X to $J_{\mathtt{GM}}$. If $X \in \mathbb{G}$ already appears in the database St_{trans}, $\mathtt{J}_{\mathtt{GM}}$ halts and returns $\perp$ to $\mathtt{J}_{\mathtt{user}}$.

2. $\mathtt{J}_{\mathtt{GM}}$ assigns to u an available leaf v of identifier $\mathtt{ID}(v)$ in the tree. The corresponding identifiers in the path from the root to v are $I_1 = 1, \ldots, I_\ell = \mathtt{ID}(v) \in \{N, \ldots, 2N-1\}$. Then $\mathtt{J}_{\mathtt{GM}}$ does the following.

   a. Commit the vector $(I_1, \ldots, I_\ell)$ as $C_v = \prod_{\kappa=1}^{\ell} g_{n+1-\kappa}^{I_\kappa}$.

   b. Using $sk^{(2)}_{\text{AHO}}$, generate an AHO signature $\sigma_{C_v} = (\hat{\theta}_1, \ldots, \hat{\theta}_7)$ on the pair $(X, C_v) \in \mathbb{G}^2$ to bind C_v to the value X that identifies u.

3. $\mathtt{J}_{\mathtt{GM}}$ sends $\mathtt{ID}(v)$ and C_v to $\mathtt{J}_{\mathtt{user}}$ that halts if either of then is found incorrect. Otherwise, $\mathtt{J}_{\mathtt{user}}$ sends a normal signature sig_u on $X||(I_1, \ldots, I_\ell)$.

4. $\mathtt{J}_{\mathtt{GM}}$ checks if the signature is valid. If not, $\mathtt{J}_{\mathtt{GM}}$ aborts. Otherwise, $\mathtt{J}_{\mathtt{GM}}$ returns σ_{C_v} to $\mathtt{J}_{\mathtt{user}}$ and stores $\mathtt{transcript}_u = (X, C_v, \sigma_{C_v}, sig_u)$ in St_{trans}.

5. $\mathtt{J}_{\mathtt{user}}$ defines the membership certificate as $\mathtt{cert}_u = (\mathtt{ID}(v), X, C_v, \sigma_{C_v}) \in \{N, \ldots, 2N-1\} \times \mathbb{G}^9$. The membership secret $\mathtt{sec}_u$ is defined as $\mathtt{sec}_u = x \in \mathbb{Z}_p$.

Revoke($\mathcal{Y}, \mathcal{S}_{\mathtt{GM}}, t, \mathcal{R}_t$): Parse $\mathcal{S}_{\mathtt{GM}} := \{sk^{(d)}_{\text{AHO}}\}_{d \in \{0,1,2,3\}}$ and do the following.

1. Using the SD method for $\mathcal{R}_t$, find a cover of unrevoked users' subsets $S_1, \ldots, S_m$ and let ϕ_i and ψ_i be the level of the primary ($\mathtt{P}_i$) and secondary ($\mathtt{S}_i$) nodes of subset S_i. The ID for these nodes are ID($\mathtt{P}_i$) and ID($\mathtt{S}_i$) respectively. Define the elements of vector of $\overrightarrow{\Phi}$, $\overrightarrow{\Psi}$, $\overrightarrow{\mathtt{P}}$, and $\overrightarrow{\mathtt{S}}$ are as follows.

$$\overrightarrow{\Phi} = (\phi_1, \ldots, \phi_i, \ldots, \phi_m) \qquad \overrightarrow{\mathtt{P}} = (\mathtt{ID}(\mathtt{P}_1), \ldots, \mathtt{ID}(\mathtt{P}_i), \ldots, \mathtt{ID}(\mathtt{P}_m))$$
$$\overrightarrow{\Psi} = (\psi_1, \ldots, \psi_i, \ldots, \psi_m) \qquad \overrightarrow{\mathtt{S}} = (\mathtt{ID}(\mathtt{S}_1), \ldots, \mathtt{ID}(\mathtt{S}_i), \ldots, \mathtt{ID}(\mathtt{S}_m))$$

2. For $\Omega = \lceil m/T \rceil$, partition $S_1, \ldots, S_m$ into Ω sequences with T elements:
$$\vec{\mathcal{S}}_1 = (S_1, \ldots, S_T), \vec{\mathcal{S}}_2 = (S_{T+1}, \ldots, S_{2T}), \ldots, \vec{\mathcal{S}}_\Omega = (S_{(\Omega-1)T+1}, \ldots, S_m).$$
The elements of $\vec{\Phi}$ is partitioned as, $\vec{\Phi}_1 = (\phi_1, \ldots, \phi_T)$, $\vec{\Phi}_2 = (\phi_{T+1}, \ldots, \phi_{2T})$, $\ldots$, $\vec{\Phi}_\Omega = (\phi_{(\Omega-1)T+1}, \ldots, \phi_m)$. Similarly, the partitions $(\vec{\Psi}_1, \ldots)$,$(\vec{\mathtt{P}}_1, \ldots)$, and $(\vec{\mathtt{S}}_1, \ldots)$ are obtained.

3. Using the vector commitment, compress the partitions of each type into:
$$C_{\vec{\Phi}_k} = \prod_{j=1}^{T} g_{n+1-j}^{\phi_{(k-1)T+j}}, \qquad C_{\vec{\mathtt{P}}_k} = \prod_{j=1}^{T} g_{n+1-j}^{\mathtt{ID}(\mathtt{P}_{(k-1)T+j})},$$
$$C_{\vec{\Psi}_k} = \prod_{j=1}^{T} g_{n+1-j}^{\psi_{(k-1)T+j}}, \qquad C_{\vec{\mathtt{S}}_k} = \prod_{j=1}^{T} g_{n+1-j}^{\mathtt{ID}(\mathtt{S}_{(k-1)T+j})},$$
for all $1 \le k \le \Omega$.

4. Using $sk_{\mathrm{AHO}}^{(3)}$, generate AHO signatures $\sigma_{\mathrm{RL_k}} = (\Theta_{k,1}, \ldots, \Theta_{k,7})$ on $(t, C_{\vec{\Phi}_k}, C_{\vec{\Psi}_k}, C_{\vec{\mathtt{P}}_k}, C_{\vec{\mathtt{S}}_k})$ for $1 \le k \le \Omega$, where $t \in \mathbb{Z}_p$ is the given epoch.

Return the revocation list
$$\mathrm{RL}_t = \left(t, \mathcal{R}_t, \vec{\Phi}, \vec{\Psi}, \vec{\mathtt{P}}, \vec{\mathtt{S}}, \{C_{\vec{\Phi}_k}, C_{\vec{\Psi}_k}, C_{\vec{\mathtt{P}}_k}, C_{\vec{\mathtt{S}}_k}, \sigma_{\mathrm{RL}_k} = (\Theta_{k,1}, \ldots, \Theta_{k,7})\}_{k=1}^{\Omega} \right).$$

Sign($\mathcal{Y}, t, \mathrm{RL}_t, \mathtt{cert}_u, \mathtt{sec}_u, M$): Return $\perp$ if $u \in \mathcal{R}_t$. Otherwise, to sign $M \in \{0,1\}^*$, generate a one-time signature key pair $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{G}_\lambda$. Parse $\mathtt{cert}_u$ as $(\mathtt{ID}(v_u), X, C_v, \sigma_{C_v}) \in \{N, \ldots, 2N-1\} \times \mathbb{G}^9$ and $\mathtt{sec}_u$ as $x \in \mathbb{Z}_p$. Do the following based on the previous scheme [4], to which step 1, 2 and 3 are added.

1. Using RL_t, find vector commitments $(C_{\vec{\Phi}_k}, C_{\vec{\Psi}_k}, C_{\vec{\mathtt{P}}_k}, C_{\vec{\mathtt{S}}_k})$ that contains subset S_i, where the user's leaf v_u lies. In commitment $C_{\vec{\Phi}_k}$, assume that the $\tilde{\jmath}$-th component includes ϕ_i, i.e., $i = (k-1)T + \tilde{\jmath}$. This is similar for $C_{\vec{\Psi}_k}, C_{\vec{\mathtt{P}}_k}$, and $C_{\vec{\mathtt{S}}_k}$. Define $\phi_{k,\tilde{\jmath}} = \phi_i, \psi_{k,\tilde{\jmath}} = \psi_i, \mathtt{P}_{k,\tilde{\jmath}} = \mathtt{P}_i, \mathtt{S}_{k,\tilde{\jmath}} = \mathtt{S}_i$. Set $(R_1, R_2, R_3, R_4, \tilde{R}_1, \tilde{R}_2) = (g_1^{\phi_{k,\tilde{\jmath}}}, g_1^{\psi_{k,\tilde{\jmath}}}, g_1^{\mathtt{ID}(\mathtt{P}_{k,\tilde{\jmath}})}, g_1^{\mathtt{ID}(\mathtt{S}_{k,\tilde{\jmath}})}, g_{\phi_{k,\tilde{\jmath}}}, g_{\psi_{k,\tilde{\jmath}}})$, and set $(\zeta_0, \zeta_{2n}, \tilde{\zeta}_0, \tilde{\zeta}_{2n}) = (g^{\mathtt{ID}(\mathtt{P}_{k,\tilde{\jmath}})}, g_{2n}^{\mathtt{ID}(\mathtt{P}_{k,\tilde{\jmath}})}, g^{\mathtt{ID}(\mathtt{S}_{k,\tilde{\jmath}})}, g_{2n}^{\mathtt{ID}(\mathtt{S}_{k,\tilde{\jmath}})})$. Compute GS commitment of all these elements. Then, calculate witnesses for the vector commitments:
$$W_{\vec{\Phi}_k} = \prod_{j=1, j \ne \tilde{\jmath}}^{T} g_{n+1-j+\tilde{\jmath}}^{\phi_{k,j}}, \qquad W_{\vec{\mathtt{P}}_k} = \prod_{j=1, j \ne \tilde{\jmath}}^{T} g_{n+1-j+\tilde{\jmath}}^{\mathtt{ID}(\mathtt{P}_{k,j})},$$
$$W_{\vec{\Psi}_k} = \prod_{j=1, j \ne \tilde{\jmath}}^{T} g_{n+1-j+\tilde{\jmath}}^{\psi_{k,j}}, \qquad W_{\vec{\mathtt{S}}_k} = \prod_{j=1, j \ne \tilde{\jmath}}^{T} g_{n+1-j+\tilde{\jmath}}^{\mathtt{ID}(\mathtt{S}_{k,j})},$$
which satisfies the following equalities.
$$e(C_{\vec{\Phi}_k}, g_{\tilde{\jmath}}) = e(R_1, g_n) \cdot e(W_{\vec{\Phi}_k}, g), \tag{4.1}$$
$$e(C_{\vec{\Psi}_k}, g_{\tilde{\jmath}}) = e(R_2, g_n) \cdot e(W_{\vec{\Psi}_k}, g), \tag{4.2}$$
$$e(C_{\vec{\mathtt{P}}_k}, g_{\tilde{\jmath}}) = e(R_3, g_n) \cdot e(W_{\vec{\mathtt{P}}_k}, g), \tag{4.3}$$
$$e(C_{\vec{\mathtt{S}}_k}, g_{\tilde{\jmath}}) = e(R_4, g_n) \cdot e(W_{\vec{\mathtt{S}}_k, g}). \tag{4.4}$$
$$e(R_3, g) = e(\zeta_0, g_1), \tag{4.5}$$
$$e(\zeta_{2n}, g) = e(\zeta_0, g_{2n}). \tag{4.6}$$
$$e(R_4, g) = e(\tilde{\zeta}_0, g_1), \tag{4.7}$$
$$e(\tilde{\zeta}_{2n}, g) = e(\tilde{\zeta}_0, g_{2n}). \tag{4.8}$$
To prove the above opening relations, commit $g_{\tilde{\jmath}}, C_{\vec{\Phi}_k}, C_{\vec{\Psi}_k}, C_{\vec{\mathtt{P}}_k}, C_{\vec{\mathtt{S}}_k}, W_{\vec{\Phi}_k}, W_{\vec{\Psi}_k}, W_{\vec{\mathtt{P}}_k}, W_{\vec{\mathtt{S}}_k}$, and compute GS proofs of (4.1)–(4.8) as $\pi_{\mathcal{S}_i}$.

2. To show that $g_{\tilde{\jmath}}$ is correct, re-randomize the AHO signature $\sigma_{g_{\tilde{\jmath}}}$ on $g_{\tilde{\jmath}}$ to $(\theta'_{\tilde{\jmath},1}, \ldots, \theta'_{\tilde{\jmath},7})$, commit $\{\theta_{\tilde{\jmath},i}\}_{i=\{1,2,5\}}$ and compute the GS proofs as follows:

$$\begin{aligned} A^{(0)} =& e(G_z^{(0)}, \theta'_{\tilde{\jmath},1}) \cdot e(G_r^{(0)}, \theta'_{\tilde{\jmath},2}) \cdot e(\theta'_{\tilde{\jmath},3}, \theta'_{\tilde{\jmath},4}) \cdot e(G^{(0)}, g_{\tilde{\jmath}}), \\ B^{(0)} =& e(H_z^{(0)}, \theta'_{\tilde{\jmath},1}) \cdot e(H_r^{(0)}, \theta'_{\tilde{\jmath},5}) \cdot e(\theta'_{\tilde{\jmath},6}, \theta'_{\tilde{\jmath},7}) \cdot e(H^{(0)}, g_{\tilde{\jmath}}). \end{aligned}$$

We denote the above proof by $\pi_{g_{\tilde{\jmath}}}$.

3. To prove that $g_1^{\phi_{k,\tilde{\jmath}}}$ (resp. $g_1^{\psi_{k,\tilde{\jmath}}}$) is mapped to $g_{\phi_{k,\tilde{\jmath}}}$ (resp. $g_{\psi_{k,\tilde{\jmath}}}$) for $\tau \in \{\phi_{k,\tilde{\jmath}}, \psi_{k,\tilde{\jmath}}\}$, re-randomize the AHO signature $\tilde{\sigma}_\tau$ on (g_1^τ, g_τ) to $(\tilde{\theta}'_{\tau,1}, \ldots, \tilde{\theta}'_{\tau,7})$, commit $\{\tilde{\theta}_{\tau,\iota}\}_{\iota=\{1,2,5\}}$, and compute the GS proofs of the following:

$$\begin{aligned} A^{(1)} =& e(G_z^{(1)}, \tilde{\theta}'_{\tau,1}) \cdot e(G_r^{(1)}, \tilde{\theta}'_{\tau,2}) \cdot e(\tilde{\theta}'_{\tau,3}, \tilde{\theta}'_{\tau,4}) \cdot e(G_1^{(1)}, g_1^\tau) \cdot e(G_2^{(1)}, g_\tau), \\ B^{(1)} =& e(H_z^{(1)}, \tilde{\theta}'_{k_\tau,1}) \cdot e(H_r^{(1)}, \tilde{\theta}'_{\tau,5}) \cdot e(\tilde{\theta}'_{\tau,6}, \tilde{\theta}'_{\tau,7}) \cdot e(H_1^{(1)}, g_1^\tau) \cdot e(H_2^{(1)}, g_\tau), \end{aligned}$$

for $\tau \in \{\phi_{k,\tilde{\jmath}}, \psi_{k,\tilde{\jmath}}\}$, where $R_1 = g^\tau$ and $\tilde{R}_1 = g_\tau$ (resp. $R_2 = g^\tau$, $\tilde{R}_2 = g_\tau$) for $\tau = \phi_{k,\tilde{\jmath}}$ (resp. $\tau = \psi_{k,\tilde{\jmath}}$). We denote the above proof by π_{lvl}.

4. To prove that the signer is an unrevoked user of subset S_i with $i = (k-1)T + \tilde{\jmath}$, commit C_v, and do the following, which is the same as [4].

   a. To show that $I_{\phi_{k,\tilde{\jmath}}} = \mathtt{ID}(\mathtt{P}_{k,\tilde{\jmath}})$, compute $W_{\phi_{k,\tilde{\jmath}}} = \prod_{l=1, l \neq \phi_{k,\tilde{\jmath}}}^{\ell} g_{n+1-l+\phi_{k,\tilde{\jmath}}}^{I_l}$ that satisfies the equality $e(C_v, g_{\phi_{k,\tilde{\jmath}}}) = e(g_1, g_n)^{I_{\phi_{k,\tilde{\jmath}}}} \cdot e(W_{\phi_{k,\tilde{\jmath}}}, g)$. Then, commit $W_{\phi_{k,\tilde{\jmath}}}$, and prove it by the GS proof of this:

   $$e(C_v, \tilde{R}_1) = e(R_3, g_n) \cdot e(W_{\phi_{k,\tilde{\jmath}}}, g).$$

   We denote the above proof by by π_{eq}.

   b. To show that $I_{\psi_{k,\tilde{\jmath}}} \neq \mathtt{ID}(\mathtt{S}_{k,\tilde{\jmath}})$, compute $W_{\psi_{k,\tilde{\jmath}}} = \prod_{l=1, l \neq \psi_{k,\tilde{\jmath}}}^{\ell} g_{n+1-l+\psi_{k,\tilde{\jmath}}}^{I_l}$ that satisfies the equality $e(C_v, g_{\psi_{k,\tilde{\jmath}}}) = e(g_1, g_n)^{I_{\psi_{k,\tilde{\jmath}}}} \cdot e(W_{\psi_{k,\tilde{\jmath}}}, g)$. Then, commit $W_{\psi_{k,\tilde{\jmath}}}$ and the group elements $(\Gamma, \hat{\zeta}_0, \hat{\zeta}_1, \hat{\zeta}_{2n}) = (g_1^{1/(I_{\psi_{k,\tilde{\jmath}}} - \mathtt{ID}(\mathtt{S}_{k,\tilde{\jmath}}))}, g^{I_{\psi_{k,\tilde{\jmath}}}}, g_1^{I_{\psi_{k,\tilde{\jmath}}}}, g_{2n}^{I_{\psi_{k,\tilde{\jmath}}}})$, and compute GS proofs of the following:

   $$\begin{aligned} & e(C_v, \tilde{R}_2) = e(\hat{\zeta}_1, g_n) \cdot e(W_{\psi_{k,\tilde{\jmath}}}, g), \\ & e(\hat{\zeta}_1 / R_4, \Gamma) = e(g_1, g_1), \\ & e(\hat{\zeta}_1, g) = e(\hat{\zeta}_0, g_1), \\ & e(\hat{\zeta}_{2n}, g) = e(\hat{\zeta}_0, g_{2n}). \end{aligned}$$

   We denote the above proof by π_{neq}.

5. To prove that $(E_1, E_2, E_3, E_4, E_5) = (t, C_{\vec{\phi}_k}, C_{\vec{\psi}_k}, C_{\mathtt{ID}(\vec{\mathtt{P}}_k)}, C_{\mathtt{ID}(\vec{\mathtt{S}}_k)})$ is derived from RL_t, re-randomize AHO signature σ_{RL_k} to $(\Theta'_1, \ldots, \Theta'_7)$ and compute the commitments to $\{\Theta_j\}_{j \in \{1,2,5\}}$. Then generate a GS proof π_{RL} proving

$$\begin{aligned} A^{(3)} =& e(G_z^{(3)}, \Theta'_1) \cdot e(G_r^{(3)}, \Theta'_2) \cdot e(\Theta'_3, \Theta'_4) \cdot e(G_1^{(3)}, g^t) \cdot \textstyle\prod_{\tau=2}^{5} e(G_\tau^{(3)}, E_\tau), \\ B^{(3)} =& e(H_z^{(3)}, \Theta'_1) \cdot e(H_r^{(3)}, \Theta'_5) \cdot e(\Theta'_6, \Theta'_7) \cdot e(H_1^{(3)}, g^t) \cdot \textstyle\prod_{\tau=2}^{5} e(G_\tau^{(3)}, E_\tau). \end{aligned}$$

6. Re-randomize AHO signature σ_{C_v} to $(\hat{\theta}'_1, \ldots, \hat{\theta}'_7)$ and compute commitments to X, $\{\hat{\theta}'_j\}_{j\in\{1,2,5\}}$. Then, generate GS proof π_{C_v} proving:

$$A^{(2)} = e(G_z^{(2)}, \hat{\theta}'_1) \cdot e(G_r^{(2)}, \hat{\theta}'_2) \cdot e(\hat{\theta}'_3, \hat{\theta}'_4) \cdot e(G_1^{(2)}, X) \cdot e(G_2^{(2)}, C_v),$$
$$B^{(2)} = e(H_z^{(2)}, \hat{\theta}'_1) \cdot e(H_r^{(2)}, \hat{\theta}'_5) \cdot e(\hat{\theta}'_6, \hat{\theta}'_7) \cdot e(H_1^{(2)}, X) \cdot e(H_2^{(2)}, C_v).$$

7. The following steps are the same as in [4]. Using $\mathtt{VK}$ as a tag (by first hashing it onto $\mathbb{Z}_p$ element), compute a tag-based encryption of X by drawing $z_1, z_2 \overset{R}{\leftarrow} \mathbb{Z}_p$ and setting $(\Upsilon_1, \Upsilon_2, \Upsilon_3, \Upsilon_4, \Upsilon_5) = (f_1^{z_1}, f_2^{z_2}, X \cdot g^{z_1+z_2}, (g^{\mathtt{VK}} \cdot U)^{z_1}, (g^{\mathtt{VK}} \cdot V)^{z_2})$.

8. Generate a GS proof that $com_X = (1, 1, X) \cdot \boldsymbol{f}_1^{w_{X,1}} \cdot \boldsymbol{f}_2^{w_{X,2}} \cdot \boldsymbol{f}_3^{w_{X,3}}$ and $(\Upsilon_1, \Upsilon_2, \Upsilon_3)$ are BBS encryption for the same value X. If we write $\boldsymbol{f}_3 = (f_{3,1}, f_{3,2}, f_{3,3})$, the GS commitment com_X can be written as $(f_1^{w_{X,1}} \cdot f_{3,1}^{w_{X,3}}, f_2^{w_{X,2}} \cdot f_{3,2}^{w_{X,3}}, X \cdot g^{w_{X,1}+w_{X,2}} \cdot f_{3,3}^{w_{X,3}})$, so that we have

$$com_X \cdot (\Upsilon_1, \Upsilon_2, \Upsilon_3)^{-1} = (f_1^{\chi_1} \cdot f_{3,1}^{\chi_3}, f_2^{\chi_2} \cdot f_{3,2}^{\chi_3}, g^{\chi_1+\chi_2} \cdot f_{3,3}^{\chi_3}) \tag{4.9}$$

with $\chi_1 = w_{\chi,1} - z_1$, $\chi_2 = w_{\chi,2} - z_2$, $\chi_3 = w_{\chi,3} - z_3$. Compute GS commitments com_{χ_j} to exponent χ_i using $\tilde{\boldsymbol{f}}$ for $j \in \{1, 2, 3\}$ and generate proofs $\{\pi_{eq-com,j}\}_{j=1}^3$ that χ_1, χ_2, χ_3 satisfy the three linear relations (4.9).

9. Compute a weakly Boneh-Boyen signature $\sigma_{\mathtt{VK}} = g^{1/(x+\mathtt{VK})}$ on $\mathtt{VK}$ and a commitment to $\sigma_{\mathtt{VK}}$. Then generate a GS proof $\pi_{\sigma_{\mathtt{VK}}}$ of the verification equation $e(\sigma_{\mathtt{VK}}, X \cdot g^{\mathtt{VK}}) = e(g, g)$.

10. Compute a one-time signature $\sigma_{ots} = \mathcal{S}(\mathtt{SK}, (M, \mathrm{RL}_t, \boldsymbol{\Upsilon} = (\Upsilon_1, \Upsilon_2, \Upsilon_3, \Upsilon_4, \Upsilon_5), \Omega, \mathbf{com}, \mathbf{\Pi}))$ where Ω consists of ι-th component for re-randomized $\sigma_{g_{\bar{j}}}, \sigma_{\phi_i}, \sigma_{\psi_i}, \sigma_{\mathrm{RL}_k}, \sigma_{C_v}$ for $\iota = 3, 4, 6, 7$. $\mathbf{com}$ consists of all commitments, and $\mathbf{\Pi}$ consists of all GS proofs.

Return the signature $\sigma = (\mathtt{VK}, \boldsymbol{\Upsilon}, \Omega, \mathbf{com}, \mathbf{\Pi}, \sigma_{ots})$.

Verify($\mathcal{Y}, t, \mathrm{RL}_t, \sigma, M$): If the one-time signature in σ is verified as invalid or if the tag-based encryption $\boldsymbol{\Upsilon}$ is not well-formed, return 0. Then, return 1 if all GS proofs are properly verified. Otherwise, return 0.

Open($\mathcal{Y}, t, \mathrm{RL}_t, \sigma, M, S_t, \mathcal{S}_{\mathtt{OA}}$): Return $\perp$ if $\mathtt{Verify}(\mathcal{Y}, t, \mathrm{RL}_t, \sigma, M)$=0. Otherwise, decrypt $\boldsymbol{\Upsilon}$ to get $\hat{X}$, and in the database S_{trans}, find a record $\langle u, \mathtt{transcript}_u = (X_u, \mathtt{ID}(v_u), C_v, \sigma_{C_v}, sig_u)\rangle$ s.t. $X_u = \hat{X}$. If no such record exist in S_{trans}, return $\perp$. Otherwise, return u.

4.4.3 Security

Here, the security of the proposed scheme is shown. We can prove the security against framing attack and the anonymity under the q-SDH and DLIN assumptions, as in [4], since the construction is similar. Thus, we only show the proof for the security against misidentification attacks.

Theorem 6. *The proposed scheme is secure against the misidentification attacks assuming that q-SFP and the n-FlexDHE problems are both hard.*

Proof. Assume that in the game of $\mathbf{Exp}_{\mathcal{A}}^{\text{mis}-\text{id}}(\lambda)$, the adversary outputs a non-trivial signature $\sigma^\star$ that does not open to unrevoked adversarial-controlled group member. Let $\sigma^\star = (\mathtt{VK}^\star, \boldsymbol{\Upsilon}^\star, \Omega^\star, \mathbf{com}^\star, \boldsymbol{\Pi}^\star, \sigma^\star_{ots})$ denote $\mathcal{A}$'s forgery and parse $\mathbf{com}^\star$ as $\mathbf{com}^\star =(com^\star_{C_v}$, $com^\star_X$, $com^\star_{g_{\tilde{\jmath}}}$, $com^\star_{C_{\vec{\mathtt{\Phi}}_k}}$, $com^\star_{C_{\vec{\mathtt{\Psi}}_k}}$, $com^\star_{C_{\vec{\mathtt{P}}_k}}$, $com^\star_{C_{\vec{\mathtt{S}}_k}}$, $com^\star_{W_{\vec{\mathtt{\Phi}}_k}}$, $com^\star_{W_{\vec{\mathtt{\Psi}}_k}}$, $com^\star_{W_{\vec{\mathtt{P}}_k}}$, $com^\star_{W_{\vec{\mathtt{S}}_k}}$, $\{com^\star_{R_k}\}_{k=1}^4, com^\star_{\tilde{R}_1}, com^\star_{\tilde{R}_2}, com^\star_{W_{\phi_{k,\tilde{\jmath}}}}, com^\star_{W_{\psi_{k,\tilde{\jmath}}}}, com^\star_{\Gamma}, \{com^\star_{\zeta_k}, com^\star_{\tilde{\zeta}_k}\}_{k\in\{0,2n\}}$, $\{com^\star_{\hat{\zeta}_k}\}_{k\in\{0,1,2n\}}$, $\{com^\star_{\theta'_\iota}, com^\star_{\Theta'_\iota}, com^\star_{\hat{\theta}'_\iota}, com^\star_{\tilde{\theta}'_{\tau,\iota}}\}_{\iota\in\{1,2,5\},\tau\in\{\phi_{k,\tilde{\jmath}},\psi_{k,\tilde{\jmath}}\}}$, $\{com^\star_{\chi_j}\}_{j=1}^3, com^\star_{\sigma_{\mathtt{VK}}})$.

Depending on the context of extractable commitments, we distinguish the following types.

Type I forgery: This is the case that either of the AHO signature $\sigma^\star_{\mathrm{RL}_k}$ for the elements of RL $(t^\star, C^\star_{\vec{\mathtt{\Phi}}_k}, C^\star_{\vec{\mathtt{\Psi}}_k}, C^\star_{\vec{\mathtt{P}}_k}, C^\star_{\vec{\mathtt{S}}_k})$, AHO signature $\sigma^\star_{C_v}$ for the elements of user's membership $(X^\star, C^\star_v)$, AHO signature $\sigma^\star_{g_{\tilde{\jmath}}}$ for the element $g^\star_{\tilde{\jmath}}$, or the AHO signatures $(\sigma^\star_{\phi_{k,\tilde{\jmath}}}, \sigma^\star_{\psi_{k,\tilde{\jmath}}})$ for the pair (g_τ, g_1^τ), $\tau \in \{\phi^\star_{k,\tilde{\jmath}}, \psi^\star_{k,\tilde{\jmath}}\}$ was not issued by **Setup** and oracles. As in [4], if any of the signatures is forged, it means a forgery against the AHO signature, which contradicts the q-SFP assumption. Thus we omit the proof for this forgery.

Type II forgeries: These are the cases that all of the signatures were issued. Since the signature $\sigma^\star_{\mathrm{RL}_k}$ on $(t^\star, C^\star_{\vec{\mathtt{\Phi}}_k}, C^\star_{\vec{\mathtt{\Psi}}_k}, C^\star_{\vec{\mathtt{P}}_k}, C^\star_{\vec{\mathtt{S}}_k})$ is an entry in the latest revocation list $\mathrm{RL}_{t^\star}$ on epoch $t^\star$, it is ensured that the vector commitments contain the k-th blocks compressed elements of SD trees in $\mathcal{R}_{t^\star}$. Also, due to the signature $\sigma^\star_{C_v}$ on $(X^\star, C^\star_v)$, it is ensured that $C^\star_v$ contains the correct $I^\star_1, \ldots, I^\star_l$ for user u. Furthermore, due to the GS proofs, the extracted committed values $g^\star_{\tilde{\jmath}}, R^\star_1, R^\star_2, R^\star_3, R^\star_4, C^\star_{\vec{\mathtt{\Phi}}_k}, C^\star_{\vec{\mathtt{\Psi}}_k}, C^\star_{\vec{\mathtt{P}}_k}, C^\star_{\vec{\mathtt{S}}_k}$, $W^\star_{\vec{\mathtt{\Phi}}_k}$, $W^\star_{\vec{\mathtt{\Psi}}_k}$, $W^\star_{\vec{\mathtt{P}}_k}, W^\star_{\vec{\mathtt{S}}_k}, {\zeta_0}^\star, {\zeta_{2n}}^\star, \tilde{\zeta}^\star_0, \tilde{\zeta}^\star_{2n}$ satisfy the relations,

$$e(C^\star_{\vec{\mathtt{\Phi}}_k}, g^\star_{\tilde{\jmath}}) = e(R^\star_1, g_n) \cdot e(W^\star_{\vec{\mathtt{\Phi}}_k}, g), \tag{4.10}$$
$$e(C^\star_{\vec{\mathtt{\Psi}}_k}, g^\star_{\tilde{\jmath}}) = e(R^\star_2, g_n) \cdot e(W^\star_{\vec{\mathtt{\Psi}}_k}, g), \tag{4.11}$$
$$e(C^\star_{\vec{\mathtt{P}}_k}, g^\star_{\tilde{\jmath}}) = e(R^\star_3, g_n) \cdot e(W^\star_{\vec{\mathtt{P}}_k}, g), \tag{4.12}$$
$$e(C^\star_{\vec{\mathtt{S}}_k}, g^\star_{\tilde{\jmath}}) = e(R^\star_4, g_n) \cdot e(W^\star_{\vec{\mathtt{S}}_k}, g). \tag{4.13}$$
$$e(R^\star_3, g) = e({\zeta_0}^\star, g_1), \tag{4.14}$$
$$e({\zeta_{2n}}^\star, g) = e({\zeta_0}^\star, g_{2n}). \tag{4.15}$$
$$e(R^\star_4, g) = e(\tilde{\zeta}^\star_0, g_1), \tag{4.16}$$
$$e(\tilde{\zeta}^\star_{2n}, g) = e(\tilde{\zeta}^\star_0, g_{2n}). \tag{4.17}$$

In addition, the extracted committed values $C^\star_v$, $W^\star_{\phi_{k,\tilde{\jmath}}}, W^\star_{\psi_{k,\tilde{\jmath}}}, \tilde{R}^\star_1, \tilde{R}^\star_2, R^\star_3, R^\star_4, \Gamma^\star, \hat{\zeta}^\star_0, \hat{\zeta}^\star_1, \hat{\zeta}^\star_{2n}$ satisfy the relations

$$e(C^\star_v, \tilde{R}^\star_1) = e(R^\star_3, g_n) \cdot e(W^\star_{\phi_{k,\tilde{\jmath}}}, g), \tag{4.18}$$
$$e(C^\star_v, \tilde{R}^\star_2) = e(\hat{\zeta}^\star_1, g_n) \cdot e(W^\star_{\psi_{k,\tilde{\jmath}}}, g), \tag{4.19}$$
$$e(\hat{\zeta}^\star_1 / R^\star_4, \Gamma^\star) = e(g_1, g_1), \tag{4.20}$$
$$e(\hat{\zeta}^\star_1, g) = e(\hat{\zeta}^\star_0, g_1), \tag{4.21}$$
$$e(\hat{\zeta}^\star_{2n}, g) = e(\hat{\zeta}^\star_0, g_{2n}), \tag{4.22}$$

where $R^\star_3, R^\star_4$ are the same as the above.

In **Type II forgeries**, the winning condition of the game (i.e., $u \notin U^a \backslash \mathcal{R}_{t^\star}$) means $u \in U^a \cap \mathcal{R}_{t^\star}$. This is because none of the AHO signatures is forged in Type II forgeries, and thus u has to be an adversarially-controlled user who is issued a membership certificate in Q_{a-join}. Then, $u \in U^a \cap \mathcal{R}_{t^\star}$ means that the user u is revoked at epoch $t^\star$. This is why, as in the original proof [4], $I^\star_{\phi_{k,\tilde{\jmath}}} \neq \mathtt{ID}(\mathtt{P}_{k,\tilde{\jmath}})$ or $I^\star_{\psi_{k,\tilde{\jmath}}} = \mathtt{ID}(\mathtt{S}_{k,\tilde{\jmath}})$ must hold for the correct tuple $(\phi_{k,\tilde{\jmath}}, \psi_{k,\tilde{\jmath}}, \mathtt{ID}(\mathtt{P}_{k,\tilde{\jmath}}), \mathtt{ID}(\mathtt{S}_{k,\tilde{\jmath}}))$ s.t. $i=(k-1)T+\tilde{\jmath}$ corresponding to any subset S_i in $\mathcal{R}_{t^\star}$, and correct $I^\star_{\phi_{k,\tilde{\jmath}}}, I^\star_{\psi_{k,\tilde{\jmath}}}$ of user u. Let $k^\star$ be k s.t. $\sigma^\star_{\mathrm{RL}_k}$ ensures the k-th entry $(t^\star, C^\star_{\vec{\Phi}_k}, C^\star_{\vec{\Psi}_k}, C^\star_{\vec{\mathtt{P}}_k}, C^\star_{\vec{\mathtt{S}}_k})$ in $\mathrm{RL}_{t^\star}$. Let $\tilde{\jmath}^\star$ be $\tilde{\jmath}$ s.t. $\sigma^\star_{g_{\tilde{\jmath}}}$ ensures $g^\star_{\tilde{\jmath}}$. Then, we can consider two possibilities: The extracted commited values $R^\star_1, R^\star_2, R^\star_3, R^\star_4$ are inappropriate, i.e., $(R^\star_1, R^\star_2, R^\star_3, R^\star_4) \neq (g_1^{\phi_{k^\star,\tilde{\jmath}^\star}}, g_1^{\psi_{k^\star,\tilde{\jmath}^\star}}, g_1^{\mathtt{ID}(\mathtt{P}_{k^\star,\tilde{\jmath}^\star})}, g_1^{\mathtt{ID}(\mathtt{S}_{k^\star,\tilde{\jmath}^\star})})$ for the correct tuple $(\phi_{k^\star,\tilde{\jmath}^\star}, \psi_{k^\star,\tilde{\jmath}^\star}, \mathtt{ID}(\mathtt{P}_{k^\star,\tilde{\jmath}^\star}), \mathtt{ID}(\mathtt{S}_{k^\star,\tilde{\jmath}^\star}))$ s.t. $i^\star=(k^\star-1)T+\tilde{\jmath}^\star$ corresponding to the subset $S_{i^\star}$ in $\mathcal{R}_{t^\star}$, or the values are appropriate, i.e., $(R^\star_1, R^\star_2, R^\star_3, R^\star_4) = (g_1^{\phi_{k^\star,\tilde{\jmath}^\star}}, g_1^{\psi_{k^\star,\tilde{\jmath}^\star}}, g_1^{\mathtt{ID}(\mathtt{P}_{k^\star,\tilde{\jmath}^\star})}, g_1^{\mathtt{ID}(\mathtt{S}_{k^\star,\tilde{\jmath}^\star})})$ for the correct tuple $(\phi_{k^\star,\tilde{\jmath}^\star}, \psi_{k^\star,\tilde{\jmath}^\star}, \mathtt{ID}(\mathtt{P}_{k^\star,\tilde{\jmath}^\star}), \mathtt{ID}(\mathtt{S}_{k^\star,\tilde{\jmath}^\star}))$ of the subset $S_{i^\star}$ in $\mathcal{R}_{t^\star}$. The former is caused, when our newly adopted commitments $C^\star_{\vec{\Phi}_k}, C^\star_{\vec{\Psi}_k}, C^\star_{\vec{\mathtt{P}}_k}, C^\star_{\vec{\mathtt{S}}_k}$ are improperly opened.

Therefore, we furthermore separate Type II forgeries into two cases: (a) the improper opening of $C^\star_{\vec{\Phi}_k}, C^\star_{\vec{\Psi}_k}, C^\star_{\vec{\mathtt{P}}_k}$, or $C^\star_{\vec{\mathtt{S}}_k}$, or (b) $I^\star_{\phi_{k^\star,\tilde{\jmath}^\star}} \neq \mathtt{ID}(\mathtt{P}_{k^\star,\tilde{\jmath}^\star})$ or $I^\star_{\psi_{k^\star,\tilde{\jmath}^\star}} = \mathtt{ID}(\mathtt{S}_{k^\star,\tilde{\jmath}^\star})$ for the appropriate $R^\star_1, R^\star_2, R^\star_3, R^\star_4$. (a) is correspondent to the case of the inappropriate $R^\star_1, R^\star_2, R^\star_3, R^\star_4$ due to the improper opening of the commitments $C^\star_{\vec{\Phi}_k}, C^\star_{\vec{\Psi}_k}, C^\star_{\vec{\mathtt{P}}_k}$, or $C^\star_{\vec{\mathtt{S}}_k}$. (b) is the same as the original proof [4] where $R^\star_1, R^\star_2, R^\star_3, R^\star_4$ are appropriate but $I^\star_{\phi_{k^\star,\tilde{\jmath}^\star}}, I^\star_{\psi_{k^\star,\tilde{\jmath}^\star}}$ are not properly opened from $C^\star_v$. Note that there are not any other possibility. This is because the case of not (a) and not (b) means that $R^\star_1, R^\star_2, R^\star_3, R^\star_4$ are the appropriate, and $I^\star_{\phi_{k^\star,\tilde{\jmath}^\star}} = \mathtt{ID}(\mathtt{P}_{k^\star,\tilde{\jmath}^\star})$ and $I^\star_{\psi_{k^\star,\tilde{\jmath}^\star}} \neq \mathtt{ID}(\mathtt{S}_{k^\star,\tilde{\jmath}^\star})$ for the appropriate values, which implies that u is not revoked at epoch $t^\star$ (it contradicts $u \in U^a \cap \mathcal{R}_{t^\star}$). Then, we can construct an adversary $\mathcal{A}_{\mathrm{DHE}}$ or $\mathcal{A}_{\mathrm{FlexDHE}}$ against n-DHE or n-FlexDHE problem using $\mathcal{A}$, for case (a) and (b):

Case (a): We consider a sub-case for each of $C^\star_{\vec{\Phi}_k}$, $C^\star_{\vec{\Psi}_k}$, $C^\star_{\vec{\mathtt{P}}_k}$, $C^\star_{\vec{\mathtt{S}}_k}$.

- If the opening of vector commitment $C^\star_{\vec{\Phi}_k}$ is improper (the sub-case for $C^\star_{\vec{\Psi}_k}$ is similar), we can construct $\mathcal{A}_{\mathrm{DHE}}$ as follows. $\mathcal{A}_{\mathrm{DHE}}$ is given an n-DHE instance $(g, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n})$. It follows **Setup** to generate the secret keys and the other public keys, and gives the public keys to $\mathcal{A}$ for this sub-case. In the game, $\mathcal{A}$ can query the oracles, where it is honestly responded using the generated secret keys, and finally output $\sigma^\star$. In the game, $\mathcal{A}_{\mathrm{DHE}}$ knows the value $\phi^\star \in \{1, \ldots, \ell\}$ s.t. $R^\star_1 = g_1^{\phi^\star}$, since the valid AHO signature $\sigma^\star_{\phi_{k,\tilde{\jmath}}}$ on $R^\star_1$ is issued. Similarly, the correctness of $g^\star_{\tilde{\jmath}}$ in (4.10)–(4.13) is assured by the AHO signature $\sigma^\star_{g_{\tilde{\jmath}}}$ on $g^\star_{\tilde{\jmath}}$. In this sub-case, equation (4.10) implies

$$e(g^\star_{\tilde{\jmath}}, C^\star_{\vec{\Phi}_k}) = e(g_1, g_n)^{\phi^\star} \cdot e(g, W^\star_{\vec{\Phi}_k}). \tag{4.23}$$

Note that the improper opening means $\phi^\star \neq \phi_{k^\star,\tilde{\jmath}^\star}$. Since $\mathcal{A}_{\mathrm{DHE}}$ knows the correct $(\phi_{k^\star,1}, \ldots, \phi_{k^\star,T})$ such that $C^\star_{\vec{\Phi}_k} = \prod_{j=1}^{T} g_{n+1-j}^{\phi_{k^\star,j}}$ in the response of the latest Q_{revoke} query, he can compute $W' = \prod_{j=1, j\neq\tilde{\jmath}^\star}^{T} g_{n+1-j+\tilde{\jmath}^\star}^{\phi_{k^\star,j}}$ which satisfies

$$e(g^\star_{\tilde{\jmath}}, C^\star_{\vec{\Phi}_k}) = e(g_1, g_n)^{\phi_{k^\star,\tilde{\jmath}^\star}} \cdot e(g, W'). \tag{4.24}$$

By dividing equations (4.23) over (4.24), we obtain the equality

$$e(g_1, g_n)^{(\phi_{k^\star,\tilde{\jmath}^\star} - \phi^\star)} = e(g, W^\star_{\vec{\Phi}_k}/W')$$

and thus $g_{n+1} = (W^\star_{\vec{\Phi}_k}/W')^{1/(\phi_{k^\star,\tilde{\jmath}^\star} - \tilde{\phi})}$. This forms a solution to the n-DHE problem.

- Consider the sub-case that the opening of $C^\star_{\vec{\mathtt{P}}_k}$ is improper (the sub-case of $C^\star_{\vec{\mathtt{S}}_k}$ is similar). As in the above sub-case for $C^\star_{\vec{\Phi}_k}$, we can construct $\mathcal{A}_{\text{FlexDHE}}$, where the simulation of the game is the same. However, in the output $\sigma^\star$, $\varrho^\star = \log_{g_1} R^\star_3$ is unknown, since $R^\star_3$ is not ensured by AHO signatures. This case of improper opening means $\varrho \star \neq \mathtt{ID}(\mathtt{P}_{k^\star,\tilde{\jmath}^\star})$. For $\varrho^\star = \log_{g_1}(R^\star_3)$, equation (4.12), (4.14), (4.15) imply that

$$e(g^\star_{\tilde{\jmath}}, C^\star_{\vec{\mathtt{P}}_k}) = e(g_1, g_n)^{\varrho^\star} \cdot e(g, W^\star_{\vec{\mathtt{P}}_k}), \tag{4.25}$$
$$\zeta_0{}^\star = g^{\varrho^\star}, \tag{4.26}$$
$$\zeta_{2n}{}^\star = g_{2n}^{\varrho^\star}. \tag{4.27}$$

Also, as in the above sub-case, $\mathcal{A}_{\text{FlexDHE}}$ can compute $W' = \prod_{j=1, j\neq\tilde{\jmath}^\star}^{T} g_{n+1-j+\tilde{\jmath}^\star}^{\mathtt{ID}(\mathtt{P}_{k^\star,j})}$ such that

$$e(g^\star_{\tilde{\jmath}}, C^\star_{\vec{\mathtt{P}}_k}) = e(g_1, g_n)^{\mathtt{ID}(\mathtt{P}_{k^\star,\tilde{\jmath}^\star})} \cdot e(g, W'). \tag{4.28}$$

By dividing equation (4.25) over (4.28), we obtain the equality $e(g_1, g_n)^{\varrho^\star - \mathtt{ID}(\mathtt{P}_{k^\star,\tilde{\jmath}^\star})} = e(g, W'/W^\star_{\vec{\mathtt{P}}_k})$, and thus $W'/W^\star_{\vec{\mathtt{P}}_k} = g_{n+1}^{\varrho^\star - \mathtt{ID}(\mathtt{P}_{k^\star,\tilde{\jmath}^\star})}$. Note that g_{n+1} cannot be computed as the above sub-case, due to the unknown $\varrho^\star$. The triple

$$(\zeta_0{}^\star \cdot g^{-\mathtt{ID}(\mathtt{P}_{k^\star,\tilde{\jmath}^\star})}, W'/W^\star_{\vec{\mathtt{P}}_k}, \zeta_{2n}{}^\star \cdot g_{2n}^{-\mathtt{ID}(\mathtt{P}_{k^\star,\tilde{\jmath}^\star})}) = (g^{\varrho^\star - \mathtt{ID}(\mathtt{P}_{k^\star,\tilde{\jmath}^\star})}, g_{n+1}^{\varrho^\star - \mathtt{ID}(\mathtt{P}_{k^\star,\tilde{\jmath}^\star})}, g_{2n}^{\varrho^\star - \mathtt{ID}(\mathtt{P}_{k^\star,\tilde{\jmath}^\star})})$$

forms a non-trivial solution to the n-FlexDHE problem.

Case(b): This is the case that the opening of $C^\star_v$ is improper, and all the openings of $(C^\star_{\vec{\Phi}_k}, C^\star_{\vec{\Psi}_k}, C^\star_{\vec{\mathtt{P}}_k}, C^\star_{\vec{\mathtt{S}}_k})$ are correctly done. Thus, the latter means ($R^\star_1$, $R^\star_2$, $R^\star_3$, $R^\star_4$) $= (g_1^{\phi_{k^\star,\tilde{\jmath}^\star}}$, $g_1^{\psi_{k^\star,\tilde{\jmath}^\star}}$, $g_1^{\mathtt{ID}(\mathtt{P}_{k^\star,\tilde{\jmath}^\star})}$, $g_1^{\mathtt{ID}(\mathtt{S}_{k^\star,\tilde{\jmath}^\star})})$. Similarly to case (a), we can construct $\mathcal{A}_{\text{DHE}}$ or $\mathcal{A}_{\text{FlexDHE}}$. In the game, the AHO signatures $\sigma^\star_{\phi_{k,\tilde{\jmath}}}, \sigma^\star_{\psi_{k,\tilde{\jmath}}}$ ensure that $\tilde{R}^\star_1 = g_{\phi_{k^\star,\tilde{\jmath}^\star}}$ and $\tilde{R}^\star_2 = g_{\psi_{k^\star,\tilde{\jmath}^\star}}$. Then, for the output $\sigma^\star$ of $\mathcal{A}$ we can prove this case in the same way as [4]. Consider these two sub-cases:

- $I^\star_{\phi_{k^\star,\tilde{\jmath}^\star}} \neq \mathtt{ID}(\mathtt{P}_{k^\star,\tilde{\jmath}^\star})$: In this case, equation (4.18) implies

$$e(g_{\phi_{k^\star,\tilde{\jmath}^\star}}, C^\star_v) = e(g_1, g_n)^{\mathtt{ID}(\mathtt{P}_{k^\star,\tilde{\jmath}^\star})} \cdot e(g, W^\star_{\phi_{k,\tilde{\jmath}}}). \tag{4.29}$$

for values $\phi_{k^\star,\tilde{\jmath}^\star}$ and $\mathtt{ID}(\mathtt{P}_{k^\star,\tilde{\jmath}^\star})$ that are available to $\mathcal{A}_{\text{DHE}}$. Since it knows $(I^\star_1, \ldots, I^\star_\ell)$, it can compute $W' = \prod_{k=1, k\neq\phi_{k^\star,\tilde{\jmath}^\star}}^{\ell} g_{n+1-k+\phi_{k^\star,\tilde{\jmath}^\star}}^{I_k}$, which satisfies

$$e(g_{\phi_{k^\star,\tilde{\jmath}^\star}}, C^\star_v) = e(g_1, g_n)^{I^\star_{\phi_{k^\star,\tilde{\jmath}^\star}}} \cdot e(g, W'). \tag{4.30}$$

By combining both equations, we find that $g_{n+1} = (W^\star_{\phi_{k,\tilde{\jmath}}}/W')^{1/(I^\star_{\phi_{k^\star,\tilde{\jmath}^\star}} - \mathtt{ID}(\mathtt{P}_{k^\star,\tilde{\jmath}^\star}))}$ is computable by $\mathcal{A}_{\text{DHE}}$ and it solves an instance the n-DHE problem.

- $I^\star_{\psi_{k^\star,\tilde{\jmath}^\star}} = \mathtt{ID}(\mathtt{S}_{k^\star,\tilde{\jmath}^\star})$: In this case, we define $\varrho^\star = \log_{g_1}(\hat{\zeta}_1)$. Equation (4.19)–(4.22) imply

$$e(g_{\psi_{k^\star,\tilde{\jmath}^\star}}, C^\star_v) = e(g_1, g_n)^{\varrho^\star} \cdot e(g, W^\star_{\psi_{k,\tilde{\jmath}}}), \tag{4.31}$$
$$g^{\varrho^\star - \mathtt{ID}(\mathtt{S}_{k^\star,\tilde{\jmath}^\star})} \neq 1_{\mathbb{G}}, \tag{4.32}$$
$$\hat{\zeta}^\star_0 = g^{\varrho^\star}, \tag{4.33}$$
$$\hat{\zeta}^\star_{2n} = g_{2n}^{\varrho^\star} \tag{4.34}$$

	Group PK size	Sig size	Membership cert. size	Revocation List Size	Signing Time	Verifying Time	Revocation Time
[4]	$O(\log N)$	$O(1)$	$O(1)$	$O(R)$	$O(1)$	$O(1)$	$O(R)$
[28]	$O(T \log N)$	$O(1)$	$O(T)$	$O(R/T)$	$O(T)$	$O(1)$	$O(R \log N)$
[22]	$O(1)$	$O(1)$	$O(R')$	$O(1)$	$O(R)$	$O(1)$	$O(R)$
This work	$O(T + \log N)$	$O(1)$	$O(1)$	$O(R/T)$	$O(T)$	$O(1)$	$O(R)$

Table 4.1: Comparisons of revocable group signature schemes.

	T=16		T=49		T=100	
	[28]	This work	[28]	This work	[28]	This work
R=1,000	63KB	88KB	21KB	29KB	10KB	14KB
R=10,000	630KB	880KB	210KB	290KB	100KB	140KB
R=100,000	6,300KB	8,800KB	2,100KB	2,900KB	1,000KB	1,400KB

Table 4.2: Concrete comparisons of the revocation list size (cryptographic part).

Also, $\mathcal{A}_{\text{FlexDHE}}$ can compute

$$W' = \prod_{k=1, k \neq \psi_{k^\star, \bar{\jmath}^\star}}^{\ell} g_{n+1-k+\psi_{k^\star, \bar{\jmath}^\star}}^{I_k^\star}$$

such that

$$e(g_{\psi_{k^\star, \bar{\jmath}^\star}}, C_v^\star) = e(g_1, g_n)^{I^\star_{\psi_{k^\star, \bar{\jmath}^\star}}} \cdot e(g, W'). \tag{4.35}$$

If we divide equation (4.31) by (4.35), we obtain the equality $e(g_1, g_n)^{\varrho^\star - I^\star_{\psi_{k^\star, \bar{\jmath}^\star}}} = e(g, W'/W^\star_{\psi_{k, \bar{\jmath}}})$, so that $W'/W^\star_{\psi_{k, \bar{\jmath}}} = g_{n+1}^{\varrho^\star - I^\star_{\psi_{k^\star, \bar{\jmath}^\star}}}$. The triple

$$(\hat{\zeta}_0^\star \cdot g^{-I^\star_{\psi_{k^\star, \bar{\jmath}^\star}}}, W'/W^\star_{\psi_{k, \bar{\jmath}}}, \hat{\zeta}_{2n}^\star \cdot g_{2n}^{-I^\star_{\psi_{k^\star, \bar{\jmath}^\star}}}) = (g^{\varrho^\star - I^\star_{\psi_{k^\star, \bar{\jmath}^\star}}}, g_{n+1}^{\varrho^\star - I^\star_{\psi_{k^\star, \bar{\jmath}^\star}}}, g_{2n}^{\varrho^\star - I^\star_{\psi_{k^\star, \bar{\jmath}^\star}}})$$

thus forms a non-trivial solution to the n-FlexDHE problem.

In any of the cases, we observe that $\mathcal{A}_{\text{FlexDHE}}$ or $\mathcal{A}_{\text{DHE}}$ solves either the n-FlexDHE instance or the n-DHE problem. The n-FlexDHE assumption is the stronger assumption. This completes the proof since $\sigma^\star$ cannot constitute a successful misidentification attack without being **Type I** or **Type II** forgery. □

4.5 Efficiency Consideration

This section compares our proposed scheme to [4] and revocable group signature schemes [22], [28] with compact RL. Comparisons are given as in Table 4.1 in terms of computational costs and the size (measured by the number of group elements) of public keys, signatures, membership certificates and revocation lists. Let N be the maximum number of users, R be the number of revoked users, R' be the maximum number of revoked users, and T be the compression parameter. In the following concrete comparisons, we consider the 128-bit security level and we assume that an element of $\mathbb{G}$ has a 512-bit representation as in [4].

From Table 4.1, this work reduces the RL size of [4] to $O(R/T)$, which is the same as the scheme of [28]. To explore the differences between this work and [28], we show the

	[4]	This work(T=$\log N$)
N=10,000(R=1,000)	880KB	108KB
N=100,000(R=10,000)	8,800KB	880KB
N=1,000,000(R=100,000)	88,000KB	7,040KB

Table 4.3: Concrete comparisons of the revocation list size (cryptographic part) for $T = \log N$ setting.

concerete comparisons of RL size in Table 4.2, using the same setting as in [28]. There consist two parts in RL, the non-cryptographic part and the cryptographic part. In our proposed scheme, we define the non-cryptographic part as $\{\mathcal{R}_t, \overrightarrow{\Phi}, \overrightarrow{\Psi}, \overrightarrow{\mathtt{P}}, \overrightarrow{\mathtt{S}}\}$. This part is the same with [4] and [28]. The cryptographic part in the proposed scheme is defined as $\{C_{\vec{\Phi}_k}, C_{\vec{\Psi}_k}, C_{\vec{\mathtt{P}}_k}, C_{\vec{\mathtt{S}}_k}, \sigma_{\mathrm{RL}_k} = (\Theta_{k,1}, \ldots, \Theta_{k,7})\}_{k=1}^{\Omega}$ that needs $512 \cdot 11\Omega$ bits, where $\Omega = m/T$, which is bounded by $512 \cdot 11\lceil(2R-1)/T\rceil$. In [28], the cryptographic part is bounded by $512 \cdot 8\lceil(2R-1)/T\rceil$. Hence, we can observe that, although this work and [28] reduce the RL size to $O(R/T)$, the RL size of this work is slightly longer.

However, the advantages against [28] are the reduction from $O(T \log N)$ to $O(T + \log N)$ of the public key size, and from $O(T)$ to $O(1)$ of the membership certificate size.

In comparison with [4], if we set $T = \log N$, the public key size of this work is about the same. Additionally, in the setting, we compare the concrete RL sizes of this work and [4] in Table 4.3. From the table, we understand that this work is capable of reducing the size of RL in $O(\log N)$ public key size setting.

Compared to [4], our proposed scheme and [28] have a signing time of $O(T)$, due to the extra W computations of vector commitment or accumulator. Our scheme also has $O(1)$ overhead in the signing and verification due to extra GS proofs.

From Table 4.1, the signature size is constant for all schemes. However, to explore the concrete size differences, we compare those in [4], [28], and [22] to ours. The shortest signature of all the compared schemes is 6KB with 98 $\mathbb{G}$-elements in [22]. On the other hand, the signature size in both of the previous schemes [4] and [28] is 9KB with 144 and 143 $\mathbb{G}$-elements, respectively. Unlikely, our proposed scheme needs 19KB with 299 $\mathbb{G}$-elements, which is about twice longer than that of [4] and [28], due to the extra GS proofs.

The scheme [22] has a constant size RL size and public key size, but the scheme has a $O(R)$ signing time and $O(R')$ membership certificate size. We consider that our proposed scheme is more efficient in the mobile environment, due to lower user computation time and storage.

4.6 Conclusion

In this chapter, we have proposed a revocable group signature scheme with a compact revocation list by $O(1/T)$ compared to the previous scheme [4], where the vector commitments compress the data in the revocation list. We also show that the proposed scheme is capable of reducing the size of RL in $O(\log N)$ public key size setting, which is the same setting as [4]. As a trade-off, the signing time is increase to $O(T)$.

Chapter 5

Implementation of Revocable Group Signatures with Compact Revocation List Using Vector Commitments

5.1 Introduction

In the previous chapter, a scheme with a compact RL to reduce the RL size in [4] was proposed. In this scheme, the subsets are partitioned into a number of blocks and compressed using a *vector commitment* [2]. Compared to the accumulator in [28], this compression method is simpler. Thus, the RL size is reduced to $O(R/T)$, while the public key size is reduced to $O(T + \log N)$, and the membership certificate size is $O(1)$. However, the signing cost is $O(T)$, and the verification has constant overhead costs, since there are more proofs of the zero-knowledge fashion. This scheme seems to be practical on the RL size, but the practicality of the signing time for concrete values of T, and the overheads in the verification time are unknown, since the scheme has not been implemented.

In this chapter, we implemented the revocable scheme [26] with a pairing library in [20] where the pairing can be computed fast using "Cross-twisted χ-based Ate (Xt-Xate) pairing" on the elliptic curve. We also adopt the asymmetric pairing due to the fastness computation at 128-bit security. In addition, we make an arrangement to the construction of [26] by swaping some parameters of $\mathbb{G}_1$ and $\mathbb{G}_2$ for lower computation costs.

To evaluate the efficiency of the implemented scheme, we measured the RL size and compare it with [4]. From the result, we can see that the RL size of [4] is greatly reduced in the implemented scheme. However, as mentioned in [26], the signing time is bound to the number of compression T as a trade-off for the compact RL size. To evaluate the practicality of the implemented scheme, we also measured the signing time and a verification time in a usual PC. From the experimental result, the signing time is less than 500 ms for $T = 400$, but the verification time is about 1.5 s. Although the verification time has a relatively large constant overhead, we consider the implemented scheme is practical in a mobile environment where a user has lower computation time and storage, but the verifier is a powerful server.

5.2 Conversion for Efficient Implementation

In the previous chapter, the scheme is described using the symmetric pairing, where any element in $\mathbb{G}$ can be both the first and the second input of e. In this chapter, for efficient implementation, we convert the construction of the implemented scheme to one using the asymmetric pairing. Since $\mathbb{G}_1$ and $\mathbb{G}_2$ are distinct, elements used in the first (resp., second) input of e are generated from $\mathbb{G}_1$ (resp., $\mathbb{G}_2$).

In the conversion, as mentioned in Section II, the elements $u = g^a \in \mathbb{G}_1$, $v = \tilde{g}^b \in \mathbb{G}_2$ for $e(u, v)$ can be swapped as $v = g^b \in \mathbb{G}_1$, $u = \tilde{g}^a \in \mathbb{G}_2$ for $e(v, u)$. We consider two reasons to swap the input elements in pairing calculations. Firstly, in the utilized pairing library, elements generated from $\mathbb{G}_1$ and $\mathbb{G}_2$ have different sizes. The element of $\mathbb{G}_1$ is about 256-bit long, while the element of $\mathbb{G}_2$ is about twice longer. Therefore, the computation on $\mathbb{G}_1$ is lighter than $\mathbb{G}_2$ on the scalar multiplication calculation needed in a vector commitment method. Due to that, in the converted scheme, we compute vector commitment $C = \prod_{\kappa=1}^{T} g_{n+1-\kappa}^{m_\kappa}$ and the witness $W_j = \prod_{\kappa=1,\kappa\neq j}^{T} g_{n+1-\kappa+j}^{m_\kappa}$ in $\mathbb{G}_1$, whose costs depend on the number of compressed elements, T, and then the opening relation is arranged to be $e(C, \tilde{g}_j) = e(g_n, \tilde{g}_1)^{m_j} \cdot e(W_j, \tilde{g})$. In the later evaluation based on implementation (Section 5.3.2), we show the effectiveness of the swap by the experimental result.

Secondly, we need to swap elements of the first and second input of e in some pairing equations, due to the conversion from GS proofs for quadratic equations to linear equations. In the verification of GS proofs, the computational cost of linear equation is lighter than the cost of quadratic one. In Section 5.3.2, we show the experimental results of the computation times for linear and quadratic ones.

In addition to the above swapping processes, we also utilize the swap version of AHO signature scheme where the signed messages are in $\mathbb{G}_1$. This is due to the swapped vector commitments signed by AHO signatures to show their correctness. Note that the above conversion to linear equations does not need the swap in AHO signatures. This is because the committed values in the converted linear equations are $\mathbb{G}_2$ elements, some values of which are signed by AHO signatures.

5.2.1 Construction of Converted Scheme

Here, we describe the summarized construction of the converted group signature scheme, as follows.
Setup: Given the number of compression T and the depth of the binary tree ℓ, this algorithm generates

1. Four key pairs of AHO signatures $\{sk_{\mathrm{AHO}}^{(d)}, pk_{\mathrm{AHO}}^{(d)}\}_{d=0}^{3}$,
2. The public parameters of the vector commitment $pk_{\mathtt{VC}}$,
3. The key pair of tag-based encryption,
4. CRS of GS proofs,
5. AHO signature on messages $\tilde{g}_j$ as σ_{g_j} for $1 \le j \le T$,
6. AHO signature on messages (g_τ, g_1^τ) as $\tilde{\sigma}_\tau$ for $1 \le \tau \le \ell$.

The public key of the group signature are $\{pk^{(d)}_{\mathrm{AHO}}\}^3_{d=0}$, $pk_{\mathtt{VC}}$, the public key of tag-based encryption, the CRS, σ_{g_j}, $\tilde{\sigma}_{\tau}$. The secret key of GM is $\{sk^{(d)}_{\mathrm{AHO}}\}^3_{d=0}$, and the secret key of the opener is the secret key of tag-based encryption.
Join: In this protocol, GM issues a joining user a membership certificate.

1. The user sends $X = g^x$ for $x \overset{R}{\leftarrow} \mathbb{Z}_p$ to GM.

2. GM returns the membership certificate that is an AHO signature σ_{C_v} on (X, C_v) where $C_v = \prod_{\kappa=1}^{\ell} g_{n+1-\kappa}^{I_\kappa}$, which is the vector commitment for the user's ID path, vector $(I_1, \ldots, I_\ell)$.

Revoke: GM generates the RL on epoch t for SD subset $S_1, \ldots, S_m$ where S_i contains elements $(\phi_i, \psi_i, \mathtt{ID}(\mathtt{P}_i), \mathtt{ID}(\mathtt{S}_i))$. The RL is compressed as follows.

1. For $\Omega{=}\lceil m/T \rceil$, partition $S_1, \ldots, S_m$ into Ω sequences with T elements: $(S_1, \ldots, S_T)$, $(S_{T+1}, \ldots, S_{2T})$, ..., $(S_{(\Omega-1)T+1}, \ldots, S_m)$. The elements of $\overrightarrow{\Phi}{=}(\phi_1, \ldots, \phi_m)$ is partitioned as, $\overrightarrow{\Phi}_1{=}(\phi_1, \ldots, \phi_T)$, $\overrightarrow{\Phi}_2{=}(\phi_{T+1}, \ldots, \phi_{2T})$,.., $\overrightarrow{\Phi}_\Omega{=}$ $(\phi_{(\Omega-1)T+1}, \ldots, \phi_m)$. Similarly, the partitions $(\overrightarrow{\Psi}_1, \ldots)$, $(\overrightarrow{\mathtt{P}}_1, \ldots)$, and $(\overrightarrow{\mathtt{S}}_1, \ldots)$ are obtained.

2. Using vector commitment, compress the partitions of vector $\overrightarrow{\Phi}$ into $C_{\vec{\Phi}_k} = \prod_{j=1}^{T} g_{n+1-j}^{\phi_{(k-1)T+j}}$ for $1 \le k \le \Omega$, and do the same for $\overrightarrow{\Psi}$, $\overrightarrow{\mathtt{P}}$, $\overrightarrow{\mathtt{S}}$.

3. Generate AHO signatures $\sigma_{\mathrm{RL_k}}$ on $(t, C_{\vec{\Phi}_k}, C_{\vec{\Psi}_k}, C_{\vec{\mathtt{P}}_k}, C_{\vec{\mathtt{S}}_k})$ for $1 \le k \le \Omega$.

Return the revocation list
$$\mathrm{RL}_t = \Big(\ t, \overrightarrow{\Phi}, \overrightarrow{\Psi}, \overrightarrow{\mathtt{P}}, \overrightarrow{\mathtt{S}}, \{C_{\vec{\Phi}_k}, C_{\vec{\Psi}_k}, C_{\vec{\mathtt{P}}_k}, C_{\vec{\mathtt{S}}_k}, \sigma_{\mathrm{RL}_k}\}^{\Omega}_{k=1}\ \Big).$$

Sign: Given a message M to be signed. The signer generates a key pair $(\mathrm{SK}, \mathrm{VK})$ of one time signature and do the following.

1. Retrieve vector commitments $(C_{\vec{\Phi}_k}, C_{\vec{\Psi}_k}, C_{\vec{\mathtt{P}}_k}, C_{\vec{\mathtt{S}}_k})$ for the subset S_i including the signer. Assume that the $\tilde{\jmath}$-th coordinate in the commitments are $(\phi_i, \psi_i, \mathtt{ID}(\mathtt{P}_i), \mathtt{ID}(\mathtt{S}_i))$ of the subset S_i which are denoted as $(\phi_{k,\tilde{\jmath}}, \psi_{k,\tilde{\jmath}}, \mathtt{ID}(\mathtt{P}_{k,\tilde{\jmath}}), \mathtt{ID}(\mathtt{S}_{k,\tilde{\jmath}}))$.

2. Open $C_{\vec{\Phi}_k}$ at coordinate $\tilde{\jmath}$ by setting $R{=}\tilde{g}_1^{\phi_{k,\tilde{\jmath}}}$ and calculate witness $W_{\vec{\Phi}_k}$ which satisfies the opening relation
$$e(\underline{C_{\vec{\Phi}_k}}, \underline{\tilde{g}_{\tilde{\jmath}}}) = e(\underline{R}, \tilde{g}_n) \cdot e(\underline{W_{\vec{\Phi}_k}}, \tilde{g}), \tag{5.1}$$
where $g_{\tilde{\jmath}}$ indicates that the $\tilde{\jmath}$-th coordinate in vector committed in $C_{\vec{\Phi}_k}$ is $\phi_{k,\tilde{\jmath}}$. The underlines in the equation show committed values to be proved in GS proof. Equation (5.1) is quadratic, because it has committed values both in $\mathbb{G}_1$ and $\mathbb{G}_2$. Note that vector commitment $C_{\vec{\Phi}_k}$ and its witness $W_{\vec{\Phi}_k}$ are in $\mathbb{G}_1$. The same process is done for $C_{\vec{\Psi}_k}$.

3. Open $C_{\vec{\mathtt{P}}_k}$ at coordinate $\tilde{\jmath}$ by setting $R{=}g_1^{\mathtt{ID}(\mathtt{P}_{k,\tilde{\jmath}})}$, $\zeta_0 = g^{\mathtt{ID}(\mathtt{P}_{k,\tilde{\jmath}})}$, $\zeta_{2n} = g_{2n}^{\mathtt{ID}(\mathtt{P}_{k,\tilde{\jmath}})}$ and calculate witness $W_{\vec{\mathtt{P}}_k}$ which satisfies the opening relation
$$e(\underline{C_{\vec{\mathtt{P}}_k}}, \underline{g_{\tilde{\jmath}}}) = e(g_n, \underline{R}) \cdot e(\underline{W_{\vec{\mathtt{P}}_k}}, \tilde{g}), \tag{5.2}$$
$$e(g, \underline{R}) = e(g_1, \underline{\zeta_0}), \tag{5.3}$$
$$e(g, \underline{\zeta_{2n}}) = e(g_{2n}, \underline{\zeta_0}), \tag{5.4}$$

where equations (5.3),(5.4) are introduced for the security proof in [26] based on the n-FlexDHE assumption. Equation (5.2) is a quadratic and equations (5.3)−(5.4) are linear, where ζ_0 and ζ_{2n} in equations (4)−(5) were swapped to $\mathbb{G}_2$ to make it linear. The same process is done for $C_{\vec{s}_k}$.

4. To prove that the signer is an unrevoked user, show that $I_{\phi_{k,\tilde{\jmath}}}=\mathtt{ID}(\mathtt{P}_{k,\tilde{\jmath}})$ and $I_{\psi_{k,\tilde{\jmath}}}\neq\mathtt{ID}(\mathtt{S}_{k,\tilde{\jmath}})$ using the opening relation of the signer's C_v on coordinate $\phi_{k,\tilde{\jmath}}$ and $\psi_{k,\tilde{\jmath}}$. These process are the same as in [4], which includes equations that are similar to equations (5.3)−(5.4), where the elements in e are swapped.

5. Compute the GS proof proving the followings.
   - The opening relations in step 2) − 4) .
   - The verification for AHO signature $\sigma_{g_{\tilde{\jmath}}}$ of the opening coordinate $\tilde{g}_{\tilde{\jmath}}$.
   - The verification for AHO signature $\tilde{\sigma}_\tau$ of $\tau \in \{\phi_{k,\tilde{\jmath}}, \psi_{k,\tilde{\jmath}}\}$ to ensure the compatibility of (g_τ, g_1^τ).
   - The verification for swapped AHO signature σ_{RL_k} of RL_t.
   - The verification for swapped AHO signature σ_{C_v} for signer's membership certificate.

   AHO signatures σ_{RL_k} and σ_{C_v} are swapped, because the signed messages contain vector commitments that are computed in $\mathbb{G}_1$ for efficiency.

6. The tag-based encryption of X and BB signature, and the one-time signature are executed in the same way as [4] together with the GS proofs, where none of the elements is needed to be swapped.

7. Return the group signature σ of the commitments, GS proofs, tag-based encryption, and one-time signature.

Verify: Check the one-time signature of M and all GS proofs in the group signature σ.
Open: Return $\perp$ if **Verify**=0 for the group signature σ. Otherwise, decrypt tag-based encryption in σ to get $\hat{X}$, and find the linked user in the database of **Join**.

5.3 Implementation

5.3.1 Pairing Library

The pairing and other operations can be implemented on an elliptic curve. The pairing calculation is critical, since the pairing calculation requires much larger computational time than other calculations. Thus, we need the fast library computing the asymmetric pairing together with the underlying elliptic curve operations. We utilize the library based on "Cross-twisted χ-based Ate (Xt-Xate) pairing" [20] with 254-bit group order and the embedding degree is 12. The security level is equivalent to the 128-bit AES. The library is based on the GMP library and implemented by C language due to the pursuit of the fastness.

Table 5.1: Environments of implementation and experiments.

CPU	Intel Core i5-4460(3.20GHz)
Main memory	7.8 GB
OS	Ubuntu 14.04 LTS
Multiple Precision Arithmetic Library	GMP-6.0.0
Pairing Library	ELiPS [20]
C compiler	GCC-4.8.4

5.3.2 Experiments and Evaluations

We measured the processing times and the data sizes in the implemented scheme to explore the efficiency of [26]. The environments of the implementation and experiments are shown in Table 5.1. For measuring the time, we utilize gettimeofday_sec() method that is a JAVA API.

As mentioned in [26], the scheme reduces the RL size of [4] by $1/T$ where T denotes the number of compression. In Table 5.2, we show the comparison of concrete RL size in [4] and our implemented scheme of [26] while changing T, where R is 1,000, 10,000, and 100,000. From the results, the RL size of implemented scheme is getting smaller by increasing T. [4] suffers from large size of RL when the number of revocation R increases. On the other hand, the implemented scheme greatly reduces the RL.

Table 5.2: RL size on different T (cryptographic part).

	[4]	Implemented Scheme [26]		
		T=50	T=100	T=400
R=1,000	880KB	27KB	14KB	3KB
R=10,000	8,800KB	270KB	140KB	30KB
R=100,000	88,000KB	2,700KB	1,400KB	300KB

Meanwhile, the signing cost of [26] depends on T as a trade-off to the compact RL size. Therefore, we measured the computation times of the signing and verification, by changing the setting of T to be 15, 50, 100, 200, 400. The measurement results are shown in Table 5.3. From the result, we can observe that the signing time is expanded when T is increased, while the verification time shows no dependency on T. In summary, we are certain that signing is sufficiently practical for T, i.e., the signing time keeps under 0.5 seconds. However, the verification is relatively large where the verification time exceeds 1 second for every case.

Table 5.3: Signing and Verification Time.

	T=15	T=50	T=100	T=200	T=400
Signing time [ms]	357	365	378	406	465
Verification time [ms]	1576	1578	1577	1576	1576

Then, we explore the details of the signing and the verification. The signing mainly consists of the commitment and proof generations of GS proofs, and W computations for the opening of vector commitments. The verification mainly consists of verifications of the GS proofs. In both, the computations concerning the GS proofs have the constant complexity for T, since the number of GS proofs does not depend on T. Therefore, the verification process is constant. On the other hand, in the W computation of vector commitments $C_{\vec{\Phi}_k}, C_{\vec{\Psi}_k}$, $C_{\vec{\mathbb{p}}_k}, C_{\vec{\mathbb{s}}_k}$ in step 1) $-$ 3) of signing, W_j is computed as,

$$W_j = \prod_{\kappa=1,\kappa\neq j}^{T} g_{n+1-\kappa+j}^{m_\kappa},$$

to open the j-th coordinate in each vector commitment. Thus, the signing time has the dependency on T.

In Table 5.4, we show the total W computation times of $C_{\vec{\Phi}_k}, C_{\vec{\Psi}_k}$, $C_{\vec{\mathbb{p}}_k}, C_{\vec{\mathbb{s}}_k}$ for T = 15, 50, 100, 200, 400 in the implementation. From the results, we can observe that the computation time increases as T becomes larger, which influences the increase of the signing time. But, the increase of the W computation time is not so large, compared to the total signing time.

Table 5.4: Computation time of W in $\mathbb{G}_1$.

	T=15	T=50	T=100	T=200	T=400
W time [ms]	2.85	11.4	24.3	52.5	111.7

Next, we examine the effectiveness of the swapping $\mathbb{G}_1$ elements and $\mathbb{G}_2$ elements in the W computation, which we mentioned in Section 5.2, by measuring the non-swap version for W computation. Table 5.5 shows the total W computation of $C_{\vec{\Phi}_k}, C_{\vec{\Psi}_k}$, $C_{\vec{\mathbb{p}}_k}, C_{\vec{\mathbb{s}}_k}$ in $\mathbb{G}_2$ for T = 15, 50, 100, 200, 400. In the comparison with Table 5.4, the W computation time in $\mathbb{G}_1$ from $\mathbb{G}_2$ is reduced on the average of 66.5%. Thus, we can confirm that swapping W to $\mathbb{G}_1$ from $\mathbb{G}_2$ in the opening relation

$$e(C, \tilde{g}_j) = e(g_1, \tilde{g}_n)^{m_j} \cdot e(W_j, \tilde{g})$$

reduces the computation cost of W in signing.

Table 5.5: Computation time of W in $\mathbb{G}_2$.

	T=15	T=50	T=100	T=200	T=400
W time [ms]	8.23	34.06	73.04	158.19	335.02

Table 5.6: Comparison of Linear and Quadratic Equations.

	Quadratic Equation	Linear Equation
Proof generation time [ms]	20.0	4.8
Verification time [ms]	117.2	33.5

Furthermore, we examine the effectiveness of converting the quadratic equation of GS proof verification to a linear equation. In Table 5.6, we show the process times of quadratic equation and linear equation for equation (5.3). The result shows that the conversion from quadratic equation to linear equation reduces approximately 70% of the proof generation and the verification process. This is because the GS proof for the linear equation generates 2 elements, while that for the quadratic equation generates 8 elements. Then, these elements are verified where the verification of linear equation includes only 4 pairing computations, while that of the quadratic equation includes 20 pairing computations. Throughout the whole construction of the implemented scheme, we had swapped 6 quadratic equations to linear equations, which means that the proof generation of this part is totally reduced to about 30 ms from about 120 ms, and the verification is reduced to about 200 ms from about 700 ms. Hence, it is clear that the swap of input elements to change the equation from quadratic to linear is an effective way to reduce the time consumption.

5.4 Conclusion

In this chapter, we implemented the converted version of revocable group signature with compact revocation list. In the conversion, we introduced two methods of swapping elements of pairing inputs in asymmetric pairing. Both methods increase the performance of the converted scheme by reducing the computation time more than 60%.

Chapter 6

Conclusion

In this thesis, we have proposed two solutions to enhance user's privacy, which are (i) an efficient anonymous credential scheme, and (ii) an efficient revocable group signature scheme. Then, we also proposed a converted version of (ii) and implemented it to show the practicality.

As the first solution, an efficient anonymous credential scheme with verifiable accumulator for monotone formula on attributes is proposed. The monotone formula is more expressive and compact than the CNF formulas, and thus the proposed system can reduce the proof generation time, compared to the previous system for CNF formulas. The size of the user's certificates is $O(2^{|U|})$ due to the restriction of the newly proposed accumulator for monotone formulas. A simple modification idea enables the reduction of the size from $O(2^{|U|})$ to $O(\sqrt{2^{|U|}})$.

As the second solution, we proposed a revocable group signature scheme with a compact revocation list by $O(1/T)$ compared to the previous scheme [4], where the vector commitments compress the data in the revocation list. We also maintain the $O(1)$ membership certificate size in [4], which is an advantage compared to [28] and [22], and the public key size is $O(T + \log N)$, which is an improvement compared to [28].

Finally, we implemented the converted version of the proposed revocable group signature with compact revocation list. In the conversion, we introduced two methods of swapping elements of pairing inputs in asymmetric pairing. Both methods increase the performance of the converted scheme by reducing the computation time more than 60%. From the experiment result, the signing time depends on the number of compression T, but even for $T = 400$, the time is under 0.5 second. Meanwhile, the verification time is exceeding 1 second and constant for every case. Although the verification time has a relatively large constant overhead, we consider that the implemented scheme is practical in a mobile environment where a user has lower computation time and storage, but the verifier is a powerful server.

Concerning the results for both solutions, we can also expect to improve them in the future. Our future work is including the applications on mobile devices in both schemes. Particularly, we need to decrease the overhead in the verification process of the proposed revocable group signature scheme. The idea of the second solution is quite clear and straightforward, which could be done in more organized structure by excluding the redundant equations and some rearrangement in the compression method. In the proposed anonymous scheme, we could work on the non-monotonic logic that shows the negative literals in the formula. This work could expand the expressiveness of the scheme and widen the practicality in many applications.

Bibliography

[1] A. Sudarsono, T. Nakanishi, and N. Funabiki, "Efficient proofs of attributes in pairing-based anonymous credential system," PETS 2011, LNCS 6794, pp.246-263, 2011.

[2] B. Libert and M. Yung, "Concise mercurial vector commitments and independent zero-knowledge sets with short proofs," TCC 2010, LNCS 5978, pp.499-517, 2010.

[3] B.Libert, T. Peters, and M. Yung, "Scalable group signatures with revocation," EUROCRYPT 2012, LNCS 7323, pp.609-627, 2012

[4] B. Libert, T. Peters, and M. Yung, "Group signatures with almost-for-free revocation, CRYPTO 2012, LNCS 7417, pp.571-589, 2012.

[5] D. Boneh and H. Shacham, "Group signatures with verifier-local revocation," ACM-CCS 2004, pp.168-177, 2004.

[6] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," CRYPTO 2004, LNCS 3152, pp. 41-55, 2004.

[7] D. Boneh and X. Boyen, "Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups," Journal of Cryptology, Vol.21, Issue 2, pp. 149-177, 2008.

[8] D. Chaum and E. Van Hejst, "Group Signature", EUROCRYPT 1991, pp. 257-265.

[9] D. Naor, M. Naor, and J. Lotspiech, "Revocation and tracing schemes for stateless receivers," CRYPTO 2001, LNCS 2139, pp.41-62, 2001.

[10] E. Bresson and J. Stern, "Group signature scheme with efficient revocation," PKC 2001, LNCS 1992, pp.190-206, 2001.

[11] E. Kiltz. "Chosen-ciphertext security from tag-based encryption," TCC 2006, LNCS 3494, pp. 198-214, 2005.

[12] J. Camenisch and A. Lysyanskaya, "Dynamic accumulators and application to efficient revocation of anonymous credentials," CRYPTO 2002, LNCS 2442, pp.61-76, 2002.

[13] J. Camenisch and A. Lysyanskaya, "An efficient system for non-transferable anonymous credentials with optional anonymity revocation," EUROCRYPT 2001, LNCS 2045, pp.93-118, 2001.

[14] J. Camenisch, M. Kohlweiss, and C. Soriente, "An accumulator based on bilinear maps and efficient revocation for anonymous credentials," PKC 2009, LNCS 5443, pp.481-432, 2009.

[15] J. Camenisch and T. Groß, “Efficient attributes for anonymous credentials,” ACM-CCS 2008, pp.345–356, 2008.

[16] J. Groth and A. Sahai, “Efficient non-interactive proof systems for bilinear groups,” EUROCRYPT 2008, LNCS 4965, pp.415-432, 2008.

[17] J. Katz, A. Sahai, B. Waters, “Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products,” Cryptology ePrint Archieve, Report 2007/404, 2007.

[18] M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev, and M. Ohkubo, “Structure-preserving signatures and commitments to group elements,” CRYPTO 2010, LNCS 6223, pp.209-236, 2010. Full Paper: Cryptology ePrint Archive, Report 2010/133.

[19] M. Abe, K. Haralambiev, and M. Ohkubo, “Signing on elements in bilinear groups for modular protocol design.” Cryptology ePrint Archieve, Report 2010/133, 2010.

[20] M. Akane, Y. Nogami, and Y. Morikawa, “Fast Ate Pairing Computation of Embedding Degree 12 Using Subfield Twisted Elliptic Curve,” IEICE Trans, Fundamentals, Vol. E92-A, No.2, pp.508-516, 2009.

[21] M. Izabachène, B. Libert, and D. Vergnaud, “Block-wise P-signatures and non-interactive anonymous credentials with efficient attributes,” IMACC 2011, LNCS 7089, pp.431-450, 2011.

[22] N. Attrapadung, K. Emura, G. Hanaoka, and Y. Sakai, “Group Signature with Constant-Size Revocation List,” The Computer Journal, Vol.58 No.10, 2015.

[23] N. Begum, T. Nakanishi, and N. Funabiki,“Efficient Proofs for CNF Formulas on Attributes in Pairing-Based Anonymous Credential System,” ICISC2012, LNCS 7839, pp. 495-509, Nov. 2012.

[24] S. Galbraith, K. Paterson, and N. Smart, ”Pairings for Cryptographers,” Technical Report. 2006, pp.165, IACR ePrint archieve, 2006.

[25] S. Sadiah, T. Nakanishi, and N. Funabiki,“A proposal of Extended Anonymous Credential Scheme with Efficient Proofs of Age Inequality,” IEICE Technical Report, ISEC2013-54, pp.21-28, Sept. 2013.

[26] S. Sadiah and T. Nakanishi ”Revocable Group Signatures with Compact Revocation List Using Vector Commitments,” WISA2016.

[27] T. Nakanishi, H. Fujii, Y. Hira, and N. Funabiki, “Revocable group signature schemes with constant costs for signing and verifying,” PKC 2009, LNCS 5443, pp.463-480, 2009.

[28] T. Nakanishi and N. Funabiki, “Revocable group signatures with compact revocation list using accumulators,” ICISC 2013, pp.435-451, 2013.

[29] T. Nakanishi and N. Funabiki, “Verifier-local revocation group sig- nature schemes with backward unlinkability from bilinear maps,” ASIACRYPT 2005, LNCS 3788, pp.533-548, 2005.