Journal of Information Processing Vol.25 949–961 (Dec. 2017)

[DOI: 10.2197/ipsjjip.25.949]

Regular Paper

Accumulator for Monotone Formulas and its Application to Anonymous Credential System

Shahidatul Sadiah[1,a)] Toru Nakanishi[1,b)] Nasima Begum[2,c)] Nobuo Funabiki[3,d)]

Received: November 29, 2016, Accepted: September 5, 2017

Abstract: An anonymous credential system allows a user to convince a service provider anonymously that he/she owns certified attributes. Previously, a system to prove AND and OR relations simultaneously by CNF formulas was proposed. To achieve a constant-size proof of the formula, this system adopts an *accumulator* that compresses multiple attributes into a single value. However, this system has a problem: the proof generation requires a large amount of computational time in the case of lots of OR literals in the formula. Therefore, we convert the attribute relation from the CNF formula to a *monotone formula* to decrease the number of OR literals. The monotone formula is a logic formula that contains any combination of AND and OR relations. In this paper, we propose an extended accumulator to prove the monotone formula, and apply it to the anonymous credential system. 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. Using this type of formula, the number of public parameters multiplied in the accumulator is decreased, which greatly impacts the reduction of authentication time.

Keywords: anonymity, anonymous credentials, accumulator, CNF formulas, monotone formulas

1. Introduction

1.1 Backgrounds

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 the user's service history, and might use it to attempt malicious activities. On the other hand, from the 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 Refs. [1], [4], [5], [12], *anonymous credential systems* were proposed, where a user can anonymously convince the 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 proves it to the SP. In the authentication, the 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 a 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.

1.2 Previous Works

In Refs. [1], [5], anonymous credential systems were proposed, where the proof of the formula has a 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 Ref. [12], a system with 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 the case of formulas with lots of OR literals.

In Ref. [14], an anonymous credential system with 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 Ref. [12], since it needs only multiplications depending on the number of OR literals. However, this system still 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.

[1] Department of Information Engineering, Hiroshima University, Higashi-Hiroshima, Hiroshima 739–8511, Japan
[2] Department of Computer Science and Engineering, University of Asia Pacific, Dhaka, Bangladesh
[3] Department of Communication Network Engineering, Okayama University, Okayama 700–8530, Japan
[a)] d152447@hiroshima-u.ac.jp
[b)] t-nakanishi@hiroshima-u.ac.jp
[c)] mail4nasima@gmail.com
[d)] funabiki@okayama-u.ac.jp

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 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.

1.3 Our Contributions

In this paper, 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. 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 [14]: The size of the user's certificate becomes exponential in the number of the user's attributes, compared to the constant size in Ref. [14] (the exponential size can be shortened as shown later). Since the scheme of Ref. [14] supports the CNF formula, by converting a monotone formula to a CNF formula, we can employ the scheme of Ref. [14] 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 Ref. [14].

The rest of this paper is organized as follows: The used cryptographic tools are explained in Section 2. Then, the extended accumulator is proposed in Section 3. The syntax and security model of the anonymous credential system are defined in Section 4, and the anonymous credential system for monotone formulas is proposed as the application of the extended accumulator in Section 5. Our proposed system is compared with the previous system [14] in Section 6, and the experimental results based on the implementation are shown in Section 7. Finally, we conclude this paper in Section 8.

Remark 1. *As in the above example, the typical case that our proposed scheme has advantage over Ref. [14] is the numerical range proof. In Ref. [13], Attribute-Based Encryption (ABE) with efficient range proofs was recently proposed, which is converted from conventional ABE by a generic method. In the converted ABE, the sublinear efficiency in the range size is achieved with expressive boolean formulas. There is some possibility that this method is applied to our setting of anonymous credentials, but the detailed investigation is one of our future works.*

2. Preliminaries

In this section, we show the cryptographic tools and proof system used as building blocks of our anonymous credential system.

2.1 Bilinear Maps

Our scheme utilizes bilinear groups and bilinear map.

(1) $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are multiplicative cyclic groups of prime order p.

(2) $g \in \mathbb{G}_1$ and $\tilde{g} \in \mathbb{G}_2$ are randomly chosen generators.

(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 $u \in \mathbb{G}_1$ and $v \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$.
- Non-degeneracy: $e(g, \tilde{g}) \neq 1_{\mathbb{G}_T}$ where $1_{\mathbb{G}_T}$ is an identity element of $\mathbb{G}_T$.

2.2 Complexity Assumptions

The security of our proposed system is based on SXDH assumption, the n-DHE (DH Exponent) assumption [6] and the q-SFP (Simultaneous Flexible Pairing) assumption [10].

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

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

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*

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

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.*

2.3 AHO Structure-Preserving Signatures

A digital signature scheme consists of algorithms **KeyGen**, **Sign**, and **Verify**. **KeyGen** generates the public and secret keys. **Sign** computes a signature on a given message using a given secret key. **Verify**, given a message and a signature together with a public key, accepts them if the signature is a valid one for the message. For a digital signature scheme, consider the following game between a challenger and the adversary:

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

(1) The challenger runs **KeyGen** to generate the public key and secret key. It gives the public key to the adversary.

(2) The adversary can adaptively choose a message, and requests the signature, and the challenger responds to the signature using **Sign**.

(3) The adversary outputs a forged signature on a message which was not requested by the adversary to the challenger.

Then, the digital signature scheme is *existentially unforgeable against chosen-message attacks*, if any PPT adversary in $\mathbf{Game}_{\mathbf{CMA}}$ can output the forged signature that is accepted by **Verify**, with negligible probability.

In previous system [14], AHO signature [9], [10] is utilized for the structure-preserving signatures, since the knowledge of the signature can be proved by Groth-Sahai proofs. Using the AHO scheme, multiple group elements are signed to obtain a constant-size signature. As in the previous construction, a single group element is signed, and thus we described in the case of single message to be signed.

AHOKeyGen: Select bilinear groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ with a prime order p and a bilinear map e. Select $G_r, H_r \in \mathbb{G}_1$ and $\mu_z, \nu_z, \mu, \nu, \alpha_a, \alpha_b \in_R \mathbb{Z}_p$. Compute $G_z = G_r^{\mu_z}, H_z = H_r^{\nu_z}, G = G_r^{\mu}, H = H_r^{\nu}, A = e(G_r, \tilde{g}^{\alpha_a}), B = e(H_r, \tilde{g}^{\alpha_b})$. Output the public key as $pk = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g, \tilde{g}, G_r, H_r, G_z, H_z, G, H, A, B)$, and the secret key as $sk = (\alpha_a, \alpha_b, \mu_z, \nu_z, \mu, \nu)$.

AHOSign: Given message $M \in \mathbb{G}_2$ together with sk, choose $\beta, \epsilon, \eta, \iota, \kappa \in_R \mathbb{Z}_p$, and compute $\theta_1 = \tilde{g}^{\beta}$, and

$$\theta_2 = \tilde{g}^{\epsilon - \mu_z \beta} M^{-\mu}, \ \theta_3 = G_r^{\eta}, \ \theta_4 = \tilde{g}^{(\alpha_a - \epsilon)/\eta},$$
$$\theta_5 = \tilde{g}^{\iota - \nu_z \beta} M^{-\nu}, \ \theta_6 = H_r^{\kappa}, \ \theta_7 = \tilde{g}^{(\alpha_b - \iota)/\kappa}.$$

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

AHOVerify: Given the message M and the signature $\sigma = (\theta_1, \ldots, \theta_7)$, accept these if the following equations are hold:

$$A = e(G_z, \theta_1) \cdot e(G_r, \theta_2) \cdot e(\theta_3, \theta_4) \cdot e(G, M),$$
$$B = e(H_z, \theta_1) \cdot e(H_r, \theta_5) \cdot e(\theta_6, \theta_7) \cdot e(H, M).$$

Under the q-SFP assumption, this signature is existentially unforgeable against chosen-message attacks [10].

Using the re-randomization algorithm, this signature can be publicly randomized to obtain another signature $(\theta'_1, \ldots, \theta'_7)$ on the same message. As a result, in the following Groth-Sahai proof, $(\theta'_i)_{i=3,4,6,7}$ can be safely revealed, while $(\theta'_i)_{i=1,2,5}$ have to be committed as mentioned in Ref. [2].

2.4 Groth-Sahai (GS) Proof

To prove the secret knowledge in relations of the bilinear maps, we utilize Groth-Sahai (GS) proofs [7]. 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$_{\mathbf{WI}}$**, $\Pr[b' = b] = 1/2$.

3. 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 [14]. The extended accumulator compresses the more general formula, the monotone formula, than CNF of the previous [14]. The construction of the accumulator is based on Ref. [14], 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 Ref. [14] is described in *intuition behind construction* in Section 3.5.

3.1 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 [14] is shown in Section 6.2). 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 [14] is also shown in Section 6.2). 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. 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 $\hat{U} = \{a_3, a_4, a_6\}$.

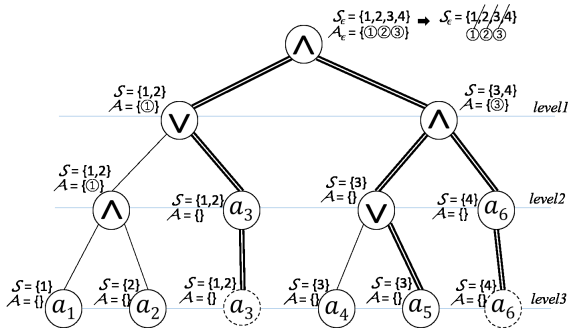

Fig. 1 Tag assignment in the tree of a monotone formula.

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 [14] in Section 6.2.

3.2 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$_{\mathbf{Acc}}$:

(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 **Game$_{\mathbf{Acc}}$** only with negligible probability.

3.3 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, $\mathcal{S}_i$ = [a..b] to each attribute i in $\mathcal{M}$, where [a..b] denotes the set of consecutive integers between a and 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 N, traverse the tree to find $T_{\mathtt{N}}$ that is the number of AND nodes in the subtree rooted by N. At the root node 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_ϵ] 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, $\mathcal{S}$=[a..b], and $\mathcal{A}$=[c..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.

ASSIGN(N,$\mathcal{S}$,$\mathcal{A}$)

if N is an OR node:

(1) Split $\mathcal{A}$ into $\mathcal{A}'$=[c..c+$T_{\mathtt{LeftChild(N)}}$-1] and $\mathcal{A}''$=[c+$T_{\mathtt{LeftChild(N)}}$..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.

if N is an AND node:

(1) Split $\mathcal{A}$ into $\mathcal{A}'$=[c..c+$T_{\mathtt{LeftChild(N)}}$-1] and $\mathcal{A}''$=[c+$T_{\mathtt{LeftChild(N)}}$+1..d], where $\mathcal{A}'$ (resp., $\mathcal{A}''$) is set as empty, if $T_{\mathtt{LeftChild(N)}}$=0 (resp., $T_{\mathtt{RightChild(N)}}$=0). In this case, one separable position a + $T_{\mathtt{LeftChild(N)}}$ is consumed in $\mathcal{A}$.

(2) Using the separate position a + $T_{\mathtt{LeftChild(N)}}$, split $\mathcal{S}'$ into $\mathcal{S}'$=[a..c+$T_{\mathtt{LeftChild(N)}}$] and $\mathcal{S}''$ = [c+$T_{\mathtt{LeftChild(N)}}$+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. 1.

In Fig. 1, $\mathcal{S}_\epsilon = \{1, 2, 3, 4\}$ and $\mathcal{A}_\epsilon$=$\{1, 2, 3\}$. The separable position t in $\mathcal{A}$ means that $\mathcal{S}$= $\{\mathtt{a}, \ldots, \mathtt{t}, \ldots, \mathtt{b}\}$ 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. 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. 1), the separable position ① is distributed to the left child with the AND descendant.

In Fig. 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 $\mathcal{S}_\epsilon = \{1, 2, 3, 4\}$.

3.4 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. 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 `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 `N*`, tag indices are divided to $\mathcal{S}'$=[a..t] and $\mathcal{S}''$=[t+1..b] at a separable position ①. Then, tag indices [t+1..b] are not assigned to leaf attributes in $\tilde{U}$ that are descendants of node `N*`. On the other hand, from ancestors of node `N*`, using other paths that do not include node `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 ① in other AND nodes, which means that any leaf except the descendants of `N*` cannot obtain [t+1..b′] for any b′. Therefore, $\{\mathcal{S}_i | i \in \tilde{U}\}$ is not a partition of $\mathcal{S}_\epsilon$. □

3.5 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 $p, \mathbb{G}_1, \mathbb{G}_2, e, g, g_1 = g^{\gamma^1}, \ldots, g_n = g^{\gamma^n}, g_{n+2} = g^{\gamma^{n+2}}, \ldots, g_{2n} = g^{\gamma^{2n}}, \tilde{g}, \tilde{g}_1 = \tilde{g}^{\gamma^1}, \ldots, \tilde{g}_n = \tilde{g}^{\gamma^n}, \tilde{g}_{n+2} = \tilde{g}^{\gamma^{n+2}}, \ldots, \tilde{g}_{2n} = \tilde{g}^{\gamma^{2n}}$ 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 $1 \leq t \leq T$. 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 [14] for CNF formulas. The only difference is how to utilize tags. In Ref. [14], 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 Ref. [14], 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 Ref. [14]. 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.

3.6 Correctness and Security of Accumulator

Based on Section 3.2, 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

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

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

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 $1 \le t \le T$, 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 $\lambda_t \le \eta$, since $\lambda_t \le |\tilde{U}|$ and η is the upper bound of $|\tilde{U}|$. Then, we have

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

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

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

Equation (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, $(1-\lambda_{\tilde{t}})c_{\tilde{t}} \le -c_{\tilde{t}}$. This is why

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

For $t \in \mathcal{T}^>$, $1-\lambda_t > 0$. However, due to $\lambda_t \ge 0$, 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

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

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

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

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 $(1-\lambda_{\tilde{t}})c_{\tilde{t}} \ge c_{\tilde{t}}$, we obtain

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

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,

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

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

From Eq. (1),

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

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 $\Delta = u - \sum_{1\le t\le T}\lambda_t c_t \neq 0$. 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. □

4. 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 [14]. Here, we show the syntax and the security model.

4.1 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.

4.2 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.

4.2.1 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.

4.2.2 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.

5. Proposed Anonymous Credential System

5.1 Construction Overview

Basically, the construction of the proposed system is similar to the previous system [14], 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 3.5. 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 $U_k (1 \le k \le K)$ 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 Ref. [14], there are two differences in construction: The first one is that the accumulator for CNF formulas in Ref. [14] is replaced by our extended accumulator for monotone formulas in Section 3.5. 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.

5.2 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:

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

(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 $acc_{\mathcal{M}}$:

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

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.

$$z^u = e(acc_{\mathcal{M}}, P_{\hat{U}}) \cdot e(g, W)^{-1}, \tag{2}$$

$$A \cdot e(\theta'_3, \theta'_4)^{-1} = e(G_z, \theta'_1) \cdot e(G_r, \theta'_2) \cdot e(G, P_{\hat{U}}), \tag{3}$$

$$B \cdot e(\theta'_6, \theta'_7)^{-1} = e(H_z, \theta'_1) \cdot e(H_r, \theta'_5) \cdot e(H, P_{\hat{U}}), \tag{4}$$

where z^u,$acc_{\mathcal{M}}$,g,A,B,θ'_3,θ'_4,θ'_6,θ'_7,G_z,G_r,G,H_z,H_r,H are public data, while $P_{\hat{U}}$,W,θ'_1,θ'_2,θ'_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 Eq. (2) shows the verification relation of accumulator:

$$\frac{e(acc_{\mathcal{M}}, P_{\hat{U}})}{e(g, W)} = z^u, \tag{5}$$

where $P_{\hat{U}} = \prod_{i \in \hat{U}} \tilde{g}_i$. Equations (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.

5.3 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 $\sigma = (\{\theta'^*_i\}_{i=3,4,6,7}, com^*_{P^*_{\tilde{U}^*}}, com^*_{W^*}, \{com^*_{\theta'^*_i}\}_{i=1,2,5}, \{\pi^*_i\}_{i=1}^3)$ 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 (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 Eqs. (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., 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 $U^{(\mathrm{i})}$ requested by $\mathcal{A}$. Respond the AHO signatures as $cert_{\mathrm{i}}$. 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.

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 $U^{(\mathrm{i})}$, 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

Table 1 Asymptotic Efficiency Comparisons.

	ProofGen Cost			**Verify** Cost			Proof	Certificate
	EXP	MUL	PAIRING	EXP	MUL	PAIRING	size	size
Previous System [14]	$O(T)$	$O(A \cdot \|U\|)$	$O(1)$	$O(T)$	$O(A)$	$O(1)$	$O(1)$	$O(1)$
Proposed System	$O(T')$	$O(A' \cdot \|\hat{U}\|)$	$O(1)$	$O(T')$	$O(A')$	$O(1)$	$O(1)$	$O(2^{\|U\|})$

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.

Proving query and the challenge query, the challenger responds using ipk and $cert_i$ 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. □

6. Comparisons and Efficiency Improvement

As mentioned in Introduction, since there are the previous systems of Refs. [1], [5], [12], [14] with constant-size attribute proofs, we briefly discuss about the comparisons. The previous systems [1], [5] support only simple AND or OR relations on attributes, and thus the proved formulas are less expressive. The previous system [12] 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 Ref. [8], 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 this paper targets, such as the example of the alcohol related website. Therefore, in the folowing detailed comparisons, we concentrate on the remaining system [14].

6.1 Efficiency Comparisons

We compare the efficiency of our proposed system to the previous system [14].

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,

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

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

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

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 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 Ref. [14] 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 Ref. [14], that of the proposed system is $O(2^{|U|})$, since $2^{|U|}$ AHO signatures are issued.

As shown in Table 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 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:
$F_1 = (Australia \vee \ldots) \wedge (1915 \vee \ldots \vee (1997 \wedge (Jan \vee \ldots \vee (Sept \wedge (1^{st} \vee \ldots \vee 5^{th}))))$.
The CNF type of formula is as follows:
$F_2 = (Australia \vee \ldots) \wedge (1915, Jan.1^{st} \vee \cdots \vee 1997, Sept.5^{th})$,
where each birthday is encoded to one attribute value such as "$1915, Jan.1^{st}$".

In the previous system [14] for CNF formulas, as described in Section 1, 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 7.3.

As indicated by the above example, the proposed system for monotone formulas is more advantageous over Ref. [14] 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 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 Ref. [14].

6.2 Comparison of Restrictions

A formula that can be used in the proposed system is in monotone formulas, while that in the previous system [14] 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 [14], 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 [14], 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 [14] does not need the assumption. As mentioned in Section 3.5, 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.

6.3 Reducing Certificate Size

As mentioned in Section 5.1, we consider that the user is issued signatures $cert = (\sigma_k)_{1 \le k \le K}$ for all subsets $U_k (1 \le k \le K)$ 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 $cert_1 = (\sigma_k)_{1 \le k \le K_1}$ and $cert_2 = (\sigma_k)_{1 \le k \le K_2}$, 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 fol-

lows:

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

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 $\sqrt{K}$.

7. 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.

7.1 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 [15]. 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" [11] 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.

7.2 Instantiation of GS Proof

For the implementation, we need to instantiate the GS proof [7] 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.

7.3 Experiment and Evaluation

To confirm the efficiency consideration of the comparison between the proposed system for monotone formulas and the previous system [14] for CNF formulas in Section 6, 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 2**.

Table 2 Environments of implementation and experiments.

CPU	Intel Core i5-4460 (3.20 GHz)
Main memory	7.8 GB
OS	Ubuntu 14.04 LTS
Multiple Precision Arithmetic Library	GMP-6.0.0
Pairing Library	ELiPS [11]
C compiler	GCC-4.8.4

Table 3 Comparison of computation times.

	Proposed System (monotone formula)	Previous System [14] (CNF formula)
Prover Time [ms]	63.04	969.11
Verifier Time [ms]	132.57	376.97

Table 4 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

Table 3 shows the processing times. The used formulas are F_1 and F_2 in Section 6, 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 4**. From this table, we can confirm that the reduction of these computation times (especially W time) influences the reduction of prover and verifier times.

8. Conclusions

In this paper, 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 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. 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 $O(\sqrt{2^{|U|}})$.

Our future work includes the applications on mobile devices, the addition of user revocation mechanism, and the application of the method of efficient range proof in ABE [13] to the anonymous credential system.

References

[1] Sudarsono, A., Nakanishi, T. and Funabiki, N.: Efficient proofs of attributes in pairing-based anonymous credential system, *Proc. 11th Privacy Enhancing Technologies Symposium* (*PETS 2011*), LNCS 6794, springer-verlag (2011).

[2] Libert, B., Peters, T. and Yung, M.: Scalable group signatures with revocation, *Advance in Cryptology–EUROCRYPT 2012*, LNCS 7323, pp.609–627, Springer-Verlag (2012).

[3] Libert, B., Peter, T., Joye, M. and Yung, M.: Linear Homomorphic Structure-Preserving Signatures and Their Applications, *Crypto 2013*, LNCS 8043, pp.289–307 (2013).

[4] Camenisch, J. and Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials, *Advances in Cryptology–CRYPTO 2002*, LNCS 2442, pp.61–76, Springer-Verlag (2002).

[5] Camenisch, J. and Groß, T.: Efficient attributes for anonymous credentials, *Proc. ACM Conference on Computer and Communications Security 2008* (*ACM-CCS'08*), pp.345–356 (2008).

[6] Camenisch, J., Kohlweiss, M. and Soriente, C.: An accumulator based on bilinear maps and efficient revocation for anonymous credentials, *Proc. 12th International Conference on Practice and Theory in Public Key Cryptography* (*PKC 2009*), LNCS 5443, pp.481–432, Springer-Verlag (2009).

[7] Groth, J. and Sahai, A.: Efficient non-interactive proof systems for bilinear groups, *Advances in Cryptology–EUROCRYPT 2008*, LNCS 4965, pp.415–432, Springer-Verlag (2008).

[8] Katz, J., Sahai, A. and Waters, B.: Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products, Cryptology ePrint Archieve, Report 2007/404 (2007).

[9] Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K. and Ohkubo, M.: Structure-preserving signatures and commitments to group elements, *Advances in Cryptology–CRYPTO 2010*, LNCS 6223, pp.209–236, Springer-Verlag (2010).

[10] Abe, M., Haralambiev, K. and Ohkubo, M.: Signing on elements in bilinear groups for modular protocol design, Cryptology ePrint Archieve, Report 2010/133 (2010).

[11] Akane, M., Nogami, Y. and Morikawa, Y.: Fast Ate Pairing Computation of Embedding Degree 12 Using Subfield Twisted Elliptic Curve, *IEICE Trans. Fundamentals* (2009).

[12] Izabachene, M., Libert, B. and Vergnaud, D.: Block-wise p-signatures and non-interactive anonymous credentials with efficient attributes, *Proc. 13th IMA Conference on Cryptography and Coding* (*IMACC2011*), LNCS 7089, Springer-Verlag (2011).
[13] Attrapadung, N., Hanaoka, G., Ogawa, K., Ohtake, G., Watanabe, H. and Yamada, S.: Attribute-Based Encryption for Range Attributes, *SCN*, pp.42–61 (2016).
[14] Begum, N., Nakanishi, T. and Funabiki, N.: Efficient Proofs for CNF Formulas on Attributes in Pairing-Based Anonymous Credential System, *IEICE Trans. Fundamentals*, Vol.E96-A, No.12, pp.2422–2433 (2013).
[15] Galbraith, S., Paterson, K. and Smart, N.: Pairings for Cryptographers, Technical Report 2006, p.165, IACR ePrint archieve (2006).
[16] Sadiah, S., Nakanishi, T. and Funabiki, N.: Anonymous Credential System with Efficient Proofs for Monotone Formulas on Attributes, *IWSEC2015*, LNCS 9241 (2015).

Shahidatul Sadiah received her B.Eng. and M.Eng. degrees in Communication Network Engineering from Okayama University, Japan, in 2013 and 2015 respectively. Currently, she is a Ph.D. candidate in the Graduate School of Engineering at Hiroshima University, Japan. Her research interests include cryptography and information security.

Toru Nakanishi received his M.S. and Ph.D. degrees in Information and Computer Sciences from Osaka University, Japan, in 1995 and 2000 respectively. He joined the Department of Information Technology at Okayama University, Japan, as a research associate in 1998, and moved to the Department of Communication Network Engineering in 2000, where he became an Assistant Professor and an Associate Professor in 2003 and 2006 respectively. In 2014, he moved to the Department of Information Engineering at Hiroshima University as a professor. His research interests include cryptography and information security. He is a member of IEICE.

Nasima Begum received his Ph.D. degree in Cryptography and Information Security from Okayama University, Japan in 2014. She received her B.Sc. and M.Sc. degrees in Computer Science and Engineering from Jahangirnagar University, Dhaka, Bangladesh, in 2006 and 2010 respectively. She joined as a Lecturer at the department of Computer Science and Engineering, Manarat International University, Dhaka, Bangladesh in January 2007. She joined as Assistant Professor at the department of Computer Science and Engineering, University of Asia Pacific, Dhaka, Bangladesh in November 2016. She has worked as a Foreign Research Fellow in Okayama University, Japan, from October 2014 to March 2016. Her research interests include cryptography and information security, signal and image processing, and artificial intelligence. She is a member of IEEE, ACM and IEICE.

Nobuo Funabiki received his B.S. and Ph.D. degrees in Mathematical Engineering and Information Physics from the University of Tokyo, Japan, in 1984 and 1993, respectively. He received his M.S. degree in Electrical Engineering from Case Western Reserve University, USA, in 1991. From 1984 to 1994, he was with the System Engineering Division, Sumitomo Metal Industries, Ltd., Japan. In 1994, he joined the Department of Information and Computer Sciences at Osaka University, Japan, as an Assistant Professor, and became an Associate Professor in 1995. He stayed at University of Illinois, Urbana-Champaign, in 1998, and at University of California, Santa Barbara, in 2000–2001, as a Visiting Researcher. In 2001, he moved to the Department of Communication Network Engineering at Okayama University as a Professor. His research interests include computer network, optimization algorithm, image processing, educational technology, Web technology, and network security. Dr. Funabiki is a member of IEEE and IEICE.