

# Dynamic Universal Accumulators for DDH Groups and Their Application to Attribute-Based Anonymous Credential Systems\*

Man Ho Au<sup>†‡</sup>   Patrick P. Tsang<sup>§¶</sup>   Willy Susilo<sup>‡</sup>   Yi Mu<sup>‡</sup>

Dartmouth Computer Science Technical Report  
TR2009-643

April 19, 2009

## Abstract

We present the first dynamic universal accumulator that allows (1) the accumulation of elements in a DDH-hard group  $\mathbb{G}$  and (2) one who knows  $x$  such that  $y = g^x$  has — or has *not* — been accumulated, where  $g$  generates  $\mathbb{G}$ , to efficiently prove her knowledge of such  $x$  in zero knowledge, and hence without revealing, e.g.,  $x$  or  $y$ .

We introduce the *Attribute-Based Anonymous Credential System* (ABACS), which allows the verifier to authenticate anonymous users according to any access control policy expressible as a formula of *possibly negated* boolean user attributes. We construct the system from our accumulator.

---

\*This paper is the extended version of the paper to appear in CT-RSA '09 under the same title.

<sup>†</sup>Corresponding author. Contact him at [mhaa456@uow.edu.au](mailto:mhaa456@uow.edu.au).

<sup>‡</sup>Centre for Computer and Information Security Research, School of Computer Science and Software Engineering, University of Wollongong, Australia

<sup>§</sup>Department of Computer Science, Dartmouth College, USA

<sup>¶</sup>Supported in part by the Institute for Security, Technology, and Society, under grant 2005-DD-BX-1091, and the National Science Foundation, under grant CNS-0524695. The views in this paper do not necessarily reflect those of the sponsors.

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Background . . . . .	3
1.2	Attribute-based anonymous credential systems . . . . .	4
1.3	Preliminaries . . . . .	6
<b>2</b>	<b>Solution Overview</b>	<b>7</b>
2.1	Dynamic universal accumulators for DDH groups . . . . .	7
2.2	Attribute-based anonymous credential systems . . . . .	7
<b>3</b>	<b>Our Dynamic Universal Accumulators for DDH Groups</b>	<b>8</b>
3.1	Definitions . . . . .	8
3.2	Constructions . . . . .	10
<b>4</b>	<b>Zero-Knowledge Protocols for Our DUA-DDH</b>	<b>13</b>
4.1	Proof of knowledge of the discrete logarithm of a committed element . . . . .	13
4.2	Proof of knowledge of a committed element in an accumulator value . . . . .	14
4.3	Proof of knowledge of a committed element not in an accumulator value . . . . .	14
<b>5</b>	<b>Our Attribute-Based Anonymous Credential System</b>	<b>15</b>
5.1	Syntax . . . . .	15
5.2	Security model . . . . .	15
5.3	Construction . . . . .	18
<b>6</b>	<b>Concluding Remarks</b>	<b>20</b>
<b>A</b>	<b>Security Proofs</b>	<b>23</b>
A.1	Proof of Theorem 1 . . . . .	23
A.2	Proof of Theorem 2 . . . . .	23
A.3	Proof of Theorem 3 . . . . .	24
<b>B</b>	<b>Instantiation of <math>PK_4</math></b>	<b>26</b>

# 1 Introduction

## 1.1 Background

### 1.1.1 Accumulators

Introduced by Benaloh and de Mare [BdM93], *accumulators* allow the representation of a set of *elements*  $Y = \{y_1, y_2, \dots, y_n\}$  by a single *value*  $v$  of size independent of  $Y$ 's cardinality; using an initial value  $u$ , one can accumulate  $Y$  into  $v$  by invoking the *accumulating function*  $f$  as  $v := f(u, Y)$ . Accumulators should be *collision-resistant* [BP97]: for any element  $y$  and any value  $v$ , there exists an efficiently computable *witness*  $w$  for  $y$  w.r.t.  $v$  *if and only if*  $y$  has been accumulated into  $v$  (often abbreviated as “ $y$  is in  $v$ ”). To prove that  $y$  is in  $v$ , one can thus demonstrate the existence of a corresponding  $w$  by proving, potentially in zero-knowledge, the knowledge of  $w$ .

Several uses of accumulators, e.g., in anonymous credential systems [CL01], require them to be *dynamic* [CL02b]: one can efficiently update an accumulator value by adding elements to — and possibly later deleting them from — the value. Furthermore, when a value is updated, e.g., from  $v$  to  $v'$ , the witness  $w$  for some element  $y$  w.r.t.  $v$  can also be efficiently updated to the witness  $w'$  for the same element  $y$  w.r.t. the new value  $v'$ . Such accumulators are called *dynamic accumulators* (DA's).

*Dynamic universal accumulators* (DUA's) [LLX07], on the other hand, are DA's with the additional property of *universality*: for any element set  $Y$  and any element  $\bar{y}$ , there exists an efficiently computable *non-membership witness*  $\bar{w}$  for  $\bar{y}$  w.r.t. value  $v = f(u, Y)$  *if and only if*  $\bar{y} \notin Y$ . By demonstrating the existence of  $\bar{w}$ , one can prove that  $\bar{y}$  is *not* in  $v$ . Non-membership witnesses should allow efficient update.

Several existing DA/DUA constructions have  $f : (u, \{y_1, y_2, \dots, y_n\}) \mapsto u^{y_1 y_2 \dots y_n} \bmod N$  as their accumulating function [BP97, CL02b, LLX07], where  $N$  is a safe-prime product and  $u \in QR(N)$ <sup>1</sup>. They permit only primes (up to a certain size) to be accumulated. Their security relies on the Strong RSA (SRSA) assumption [BP97].

Nguyen [Ngu05] constructed a DA from bilinear pairings (to be defined later). It has  $f : (u, \{y_1, y_2, \dots, y_n\}) \mapsto u^{(s+y_1)(s+y_2)\dots(s+y_n)}$  as the accumulating function, where  $s$  is the master secret of the accumulator instance and  $u$  is in some group equipped with a bilinear pairing. The construction allows elements in  $\mathbb{Z}_p \setminus \{-s\}$  for some prime  $p$  to be accumulated. Its security relies on the  $q$ -Strong Diffie-Hellman ( $q$ -SDH) assumption [BB04]. Unlike the above “SRSA-based” constructions, dynamically adding an element to a value in Nguyen's construction requires the knowledge of the master secret  $s$ .

An accumulator would not be too useful (at least for building anonymous credential systems) without a suite of efficient zero-knowledge protocols for proving various facts about the accumulator values and elements. For instance, all the aforementioned constructions come with a protocol that proves in zero-knowledge that a commitment  $c$  opens to some element in an accumulator value  $v$ .

---

<sup>1</sup> $QR(N)$  denotes the group of quadratic residues modulo  $N$ .

### 1.1.2 Anonymous credential systems

In an anonymous credential system (ACS) [CL01], those and only those users who have registered to an organization  $O$  can authenticate their membership in  $O$  to any verifier (e.g., a server, another organization, etc.) anonymously and unlinkably among the set of all members in  $O$ . Camenisch and Lysyanskaya [CL01] constructed the first ACS using a *signature scheme with efficient protocols* [CL02a] (commonly referred to as CL-signatures or P-signatures [BCKL08]) as a key building block. Many subsequent works have taken the same approach [CL02a, CL02b, CL04, BCKL08].

In this approach, to join an organization  $O$ , a user  $U$  first registers her pseudonym, which is simply a commitment of her pre-established private key  $x_U$ , e.g., in her PKI credential. Pseudonyms (even those of the same user) are hence unlinkable.  $O$  then issues a CL-signature  $\sigma_U$  on  $x_U$  according to the issuing protocol for CL-signatures, during which  $O$  learns nothing about  $x_U$ .  $U$  uses  $\sigma_U$  as her anonymous credential.

To be able to revoke membership efficiently,  $O$  can maintain a DA as a “white-list” of users whose membership has *not* yet been revoked [CL02b], by adding each user  $U$ ’s credential  $\sigma_U$  (or its identifier) to its DA when  $U$  registers and, when desired, deleting  $\sigma_U$  from DA to revoke  $U$ ’s membership. Therefore, to demonstrate her non-revoked membership in  $O$  to a verifier  $V$ ,  $U$  conducts a zero-knowledge proof that (1) she has  $O$ ’s signature on her private key, and that (2) the signature is a credential in  $O$ ’s current DA. Alternatively,  $O$  can maintain a DUA as a “blacklist” of users whose membership has been revoked [LLX07]. In this case, to demonstrate her non-revoked membership in  $O$ ,  $U$  instead proves in zero-knowledge that (1) she has  $O$ ’s signature on her private key, and that (2) the signature is *not* a credential in  $O$ ’s current DUA.

## 1.2 Attribute-based anonymous credential systems

As a major contribution of this paper, we present the *Attribute-Based Anonymous Credential System* (ABACS), which generalizes the conventional notion of anonymous credential system (ACS) [CL01], in a fashion analogous to how Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [BSW07] generalizes public-key encryption, and how attribute certificates generalize identity certificates in X.509 PKIs [FH02].

Credentials in ABACS can be more precisely referred to as *anonymous attribute credentials* — they are issued to users to certify their possession of an attribute, allowing the users to prove various facts to any verifier about their credential ownership and hence attribute possession in some anonymous fashion. ABACS thus enables *privacy-preserving attribute-based access control*, in which a server is willing to grant a user access to an object such as a file or a service so long as the attributes possessed and/or lacked by the user satisfy the server’s access control policy on the object, while privacy-concerned users desire to access the object by revealing merely the fact that they satisfy the policy, and thus concealing, e.g., their identity, how they satisfy the policy, and etc.

In this paper, we confine ourselves to *boolean attributes* only. (Some attributes such as age and weight may take a value from a wider range such as non-negative integers and real numbers, and are hence non-boolean.) Boolean attributes provide

rich semantics for labeling objects for access control. For example, they can represent group membership, or “roles” in Role-Based Access Control (RBAC).

### 1.2.1 Features

ABACS is a credential system with the following features.

- **Flexible attribute-based access control** The verifier can choose to enforce *any* access control policy expressible as a boolean attribute formula in disjunctive normal form (DNF), i.e., a disjunction of terms, where each term is a conjunction of *possibly negated* boolean attributes, e.g., “(Student  $\wedge$  Bio)  $\vee$  ( $\neg$ Bio)”.
- **Multiple ACAs** To support an attribute, a corresponding *Attribute Certification Authority* (ACA) is created (during setup or dynamically when needed) to issue credentials to users to certify their possession of that attribute. These ACAs are mutually independent; an ACA can only certify the possession of attributes for which it was created. This allows them to have different certification procedures with different trust levels, and confines the damages of their compromises.
- **Robust accountability** The verifier accepts in the authentication only if the authenticating user satisfies the access control policy being enforced, i.e., the corresponding boolean formula evaluates to **true** on input the set of attribute for which the user has acquired a credential<sup>2</sup>.

Hence, a user who has acquired a credential for an attribute can’t pretend that she hasn’t, and colluding users, none of which alone satisfy the policy, can’t satisfy it by pooling together their credentials.

- **Anonymous authentication** The verifier knows only whether an authenticating user satisfies the access control policy he is enforcing. More precisely, authentication attempts by honest users who (resp. do not) satisfy the verifier’s policy are anonymous and unlinkable among the set of all users who also (resp. do not) satisfy the policy.
- **Anonymous certification** While ACAs must make public some data related to the certification status of users’ attribute possession for authentication to be possible, some applications may require that such data reveals no (computational) information about the identity of the certified users, or more generally, no one can tell if two ACAs have issued a credential to a common user.
- **Efficiency and practical negation support** The authentication can be done in  $O(|P|)$  time, where  $|P|$  is the size of the verifier’s policy measured in the number of (negated) attributes in it, and hence regardless of, e.g., the number of users, verifiers, ACAs, or attributes that the authenticating user possesses/lacks.

---

<sup>2</sup>A (resp. negated) attribute in a formula evaluates to **true** *if and only if* it is (resp. not) contained in the user’s attribute set.

Also, a user who lacks an attribute never has to contact anyone (e.g., the corresponding ACA) before she can prove her lack of the attribute.

### 1.2.2 Applications

The two scenarios below can benefit from ABACS.

The Biology department provides free parking to its students and any visitor from outside the department. The parking lot entrance hence enforces an access control policy of “ $(\text{Bio} \wedge \text{Student}) \vee (\neg \text{Bio})$ ”. Identifiable authentication solutions<sup>3</sup> would violate the privacy desired by some users. A solution should allow different departments to locally manage their own “membership”. Also, a visitor shouldn’t have to show up at the Biology Department to get a “ $\neg \text{Bio}$ ” credential before he or she can park.

A pharmacist must check that “ $\text{Fever} \wedge \neg \text{Asthma}$ ” holds for a patient before dispensing Aspirin (as many asthma sufferers are allergic to Aspirin), while the patient may not want to disclose her entire medical record, e.g., when she has an unrelated genetic disorder. Also, a fever patient with asthma with the “help” from someone without fever or asthma must still be unable to obtain Aspirin.

### 1.3 Preliminaries

**Bilinear pairings** A bilinear pairing is a mapping from a pair of group elements to a group element. Specifically, let  $\mathbb{G}_1$  and  $\mathbb{G}_2$  be some cyclic groups of prime order  $p$ . Let  $g$  be a generator of  $\mathbb{G}_1$ . A function  $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$  is a bilinear pairing if the following holds:

- *Unique Representation.* Each element in  $\mathbb{G}_1, \mathbb{G}_2$  has unique binary representation.
- *Bilinearity.*  $e(A^x, g^y) = e(A, B)^{xy}$  for all  $A, B \in \mathbb{G}_1$  and  $x, y \in \mathbb{Z}_p$ .
- *Non-degeneracy.*  $e(g, g) \neq 1$ , where 1 is the identity element in  $\mathbb{G}_2$ .
- *Efficient Computability.*  $e(A, B)$  can be computed efficiently (i.e. in polynomial time) for all  $A, B \in \mathbb{G}_1$ .

**Complexity assumptions** The *Decisional Diffie-Hellman* (DDH) problem in  $\mathbb{G}$  is defined as follows: On input a quadruple  $(h_0, h_1, h_0^x, y^*) \in \mathbb{G}^4$ , output 1 if  $y^* = h_1^x$  and 0 otherwise. We say that the DDH assumption holds in  $\mathbb{G}$  if no PPT algorithm has non-negligible advantage over random guessing in solving the DDH problem in  $\mathbb{G}$ . We call a group *DDH-hard* if the DDH assumption holds in the group.

The *q-Strong Diffie-Hellman* ( $q$ -SDH) problem in  $\mathbb{G} = \langle g_0 \rangle$  is defined as follows: On input a  $(q+1)$ -tuple  $(g_0, g_0^\alpha, g_0^{\alpha^2}, \dots, g_0^{\alpha^q}) \in \mathbb{G}^{q+1}$ , output a pair  $(w, y) \in \mathbb{G} \times \mathbb{Z}_p^*$ , where  $p$  is the order of  $\mathbb{G}$ , such that  $w^{(\alpha+y)} = g_0$ . We say that the  $q$ -SDH assumption holds in  $\mathbb{G}$  if no PPT algorithm has non-negligible advantage in solving the  $q$ -SDH problem in  $\mathbb{G}$ .

<sup>3</sup>e.g., waving an RFID card, or an e-token installed with X.509 attribute certificates

**Zero-knowledge proof-of-knowledge** In a zero-knowledge proof of knowledge protocol [GMR89], a prover convinces a verifier that some statement is true without the verifier learning anything except the validity of the statement. We use Camenisch and Stadler’s notation [CS97]. For example,  $PK\{(x) : y = g^x\}$  denotes a zero-knowledge proof-of-knowledge protocol that proves the knowledge of the discrete logarithm of  $y$  to the base  $g$ .

## 2 Solution Overview

We briefly describe how we construct a DUA that allows the accumulation of elements in a DDH-hard group, which we call DUA-DDH. We then highlight how we build ABACS from it.

### 2.1 Dynamic universal accumulators for DDH groups

To construct DUA-DDH, we take Nguyen’s DA construction as the point of departure; we augment *universality* to it. Li et al. [LLX07] presented a technique to augment *universality* to Camenisch and Lysyanskaya’s DA construction [CL02b]. The technique, however, requires the unique factorization of integers and relies on the SRSA assumption, and hence is not immediately applicable to Nguyen’s DA. Fortunately, we make the observation that the technique works as long as the domain of accumulatable elements is (a subset of) a Euclidean domain. (In the case of Li et al.’s, the domain is the ring of integers.) Consequently, to augment *universality* to Nguyen’s construction, we adapt the technique to work on a different Euclidean domain, namely the ring of polynomials over a finite field.

We also equip our accumulator construction with a few useful zero-knowledge protocols. Of particular importance is the following pair:

$$\left\{ \begin{array}{l} PK \quad \{(x, y) : C = \text{Com}_1(x) \wedge D = \text{Com}_2(y) \wedge y = g^x \wedge y \text{ is in } v\} \\ PK \quad \{(x, y) : C = \text{Com}_1(x) \wedge D = \text{Com}_2(y) \wedge y = g^x \wedge y \text{ is not in } v\} \end{array} \right.$$

where  $\text{Com}_1$  and  $\text{Com}_2$  are commitment schemes and  $g$  generates a DDH group, the elements in which can be accumulated in our accumulator. We construct the protocol using Pedersen’s commitment scheme [Ped91] and Camenisch’s technique for proving double discrete logarithms [CS97]. The construction has a complexity of  $O(\lambda)$  for a cheating probability of  $2^{-\lambda}$ .

These protocols are the cornerstone of our ABACS construction.

### 2.2 Attribute-based anonymous credential systems

Let  $\mathbb{G}$  be a DDH group. Let ACA  $i$  be the ACA that certifies users’ possession of attribute  $i$ . Each ACA  $i$  instantiates and maintains a DUA-DDH  $A_i$  of its own, but for the same  $\mathbb{G}$ , and independently picks a generator  $g_i$  of  $\mathbb{G}$  at random.

Let  $U$  be a user with a pre-established private key  $x$ . For each attribute  $i$  she possesses, she can get certified by ACA  $i$  by providing her pseudonym  $y_i = g_i^x$  w.r.t. ACA  $i$ . ACA  $i$  then adds  $y_i$  to its  $A_i$ . To later revoke the certification, ACA  $i$  can simply delete  $y_i$  from  $A_i$ . Finally, for each attribute  $j$   $U$  lacks, she need not do anything (such as contacting ACA  $j$ ); her pseudonym w.r.t. ACA  $j$  is by default *not* in ACA  $j$ 's  $A_j$ .

Each ACA  $i$  publishes  $A_i$ ,  $g_i$  (with a proof of their correct generation) and the list of pseudonyms that have been added in  $A_i$ . Thanks to the DDH assumption, no one — not even to the ACAs — can tell which user a pseudonym belongs to, or whether two ACAs' pseudonym lists contain a common user (non-negligibly better than random guessing).

From the published information, a user can compute a (resp. non-) membership witness for each attribute  $i$  she has (resp. not) been certified. The first-time computation takes  $O(|L_i|)$  time when ACA  $i$  has certified  $|L_i|$  users. This computation can be further reduced to  $O(1)$  by moving the computation to ACA which is in possession of the auxiliary information of the accumulator. Updating the witness in the future take constant time per each change in the list of certified users.

User  $U$  who possesses attribute  $i$  and has been certified by ACA  $i$  can prove such fact to any verifier during authentication by proving that she has the knowledge of some  $x$  such that  $y_i = g_i^x$  is in  $A_i$ . Similarly, if  $U$  lacks attribute  $j$ , she can prove the fact by proving that  $y_j = g_j^x$  is not in  $A_j$ . These proofs can be accomplished in constant time. Generalizing the proof using a standard technique [CDS94], a user can prove the validity of any DNF boolean attribute formula in time linear in the size of the formula.

### 3 Our Dynamic Universal Accumulators for DDH Groups

#### 3.1 Definitions

We incrementally define *Dynamic Universal Accumulators for DDH Groups* (DUA-DDH's). We start by adapting Li et al.'s definition of *universality* to pairing-based accumulators.

**Definition 1 (Universal Accumulators (UAs))** A universal accumulator is a scheme with the following properties:

- **Efficient generation** There exists a *Probabilistic Polynomial-Time (PPT)* algorithm  $\text{Gen}$  that, on input security parameter  $1^\lambda$ , outputs a tuple  $(f, g, \mathcal{Y}_f, u, t_f)$ , where  $f$  is a function  $\mathcal{U}_f \times \mathcal{Y}'_f \rightarrow \mathcal{U}_f$  and  $g$  is another function  $\mathcal{U}_f \rightarrow \mathcal{U}_g$  for some domains  $\mathcal{Y}_f, \mathcal{U}_f, \mathcal{U}_g$ ;  $\mathcal{Y}'_f \subseteq \mathcal{Y}_f$  is the domain for accumulatable elements;  $t_f$  is some optional auxiliary information about  $f$ ; and  $u$  is an element in  $\mathcal{U}_f$ . We assume the tuple  $(f, g)$  is drawn uniformly at random from its domain.
- **Quasi-commutativity** For all  $(f, g, \mathcal{Y}_f, \cdot) \leftarrow \text{Gen}(1^\lambda)$ ,  $v \in \mathcal{U}_f$  and  $y_1, y_2 \in \mathcal{Y}'_f$ , we have  $f(f(v, y_1), y_2) = f(f(v, y_2), y_1)$ . Hence, if  $Y = \{y_1, \dots, y_k\} \subset \mathcal{Y}'_f$ , then we can denote  $f(\dots f(f(v, y_1), y_2) \dots, y_k)$  by  $f(v, Y)$  unambiguously.

- **Efficient evaluation** For all  $(f, g, \mathcal{Y}_f, t_f, u) \leftarrow \text{Gen}(1^\lambda)$ ,  $v \in \mathcal{U}_f$ , and  $Y \subset \mathcal{Y}_f$  so that  $|Y|$  is polynomial in  $\lambda$ , the function  $g \circ f(v, Y)$  is computable in time polynomial in  $\lambda$ .  $v = g \circ f(u, Y)$  represents the set  $Y$ . We call  $v$  the accumulator value for  $Y$  and say that  $y$  has been accumulated into  $v$  (or  $y$  is “in”  $v$ ), for all  $y \in Y$ .
- **Membership (resp. non-membership) witnesses** For all  $(f, g, \mathcal{Y}_f, \cdot) \leftarrow \text{Gen}(1^\lambda)$ , there exists a relation  $\Omega$  (resp.  $\bar{\Omega}$ ) that defines membership (resp. non-membership) witnesses:  $w$  (resp.  $\bar{w}$ ) is a valid membership (resp. non-membership) witness for element  $y \in \mathcal{Y}_f$  w.r.t. accumulator value  $v \in \mathcal{U}_f$  *if and only if*  $\Omega(w, y, v) = 1$  (resp.  $\bar{\Omega}(\bar{w}, y, v) = 1$ ). Membership witness (resp. non-membership witness) should be efficiently computable (in polynomial-time in  $\lambda$ ) with  $t_f$ .  $\square$

The security of universal accumulators requires that it is hard to find a valid membership (resp. non-membership) witness for an element that is *not* in (resp. is *indeed* in) an accumulator value w.r.t. that accumulator value. We employ a strong definition in which the adversary is considered successful even if he present an element that is outside the intended domain of the accumulator ( $\mathcal{Y}'_f$  instead of  $\mathcal{Y}_f$ ). Accumulators with this stronger sense of security improves efficiency of systems on which it is based because users within this system needs not conduct proof to demonstrate the elements presented is inside the intended domain of the accumulator. Below we give a precise definition.

**Definition 2 (Security of Universal Accumulators (UAs))** A universal accumulator is secure if, for any PPT algorithm  $\mathcal{A}$ , both  $P_1$  and  $P_2$  are negligible in  $\lambda$ , where:

$$\begin{cases} P_1 = \Pr \left[ (f, g, \mathcal{Y}_f, u, \cdot) \leftarrow \text{Gen}(1^\lambda); (y, w, Y) \leftarrow \mathcal{A}(g \circ f, g, \mathcal{Y}_f, u) : \right. \\ \left. Y \subset \mathcal{Y}'_f \wedge y \in \mathcal{Y}'_f \setminus Y \wedge \Omega(w, y, g \circ f(u, Y)) = 1 \right], \\ P_2 = \Pr \left[ (f, g, \mathcal{Y}_f, u) \leftarrow \text{Gen}(1^\lambda); (y, \bar{w}, Y) \leftarrow \mathcal{A}(g \circ f, g, \mathcal{Y}_f, u) : \right. \\ \left. Y \subset \mathcal{Y}'_f \wedge y \in Y \wedge \bar{\Omega}(\bar{w}, y, g \circ f(u, Y)) = 1 \right]. \end{cases}$$

$\square$

**Definition 3 (Dynamic Universal Accumulators (DUAs))** A DUA is an UA with the following additional properties:

- **Efficient update of accumulator** There exists an efficient algorithm  $D_1$  such that for all  $v = g \circ f(u, Y)$ ,  $y \notin Y$  and  $\hat{v} \leftarrow D_1(t_f, v, y)$ , we have  $\hat{v} = g \circ f(u, Y \cup \{y\})$ . If  $y \in Y$  instead, then we have  $\hat{v} = g \circ f(1, Y \setminus \{y\})$  instead.
- **Efficient update of membership witnesses** Let  $v$  and  $\hat{v}$  be the original and updated accumulator values respectively and  $\hat{y}$  be the newly added (or deleted) element. There exists an efficient algorithm  $D_2$  that, on input  $y, w, v, \hat{v}$  with  $y \neq \hat{y}$  and  $\Omega(w, y, v) = 1$ , outputs  $\hat{w}$  such that  $\Omega(\hat{w}, y, \hat{v}) = 1$ .

- **Efficient update of non-membership witnesses** Let  $v$  and  $\hat{v}$  be the original and updated accumulator values respectively and  $\hat{y}$  be the newly added (or deleted) element. There exists an efficient algorithm  $D_3$  that, on input  $y, \bar{w}, v, \hat{v}$  with  $y \neq \hat{y}$  and  $\bar{\Omega}(\bar{w}, y, v) = 1$ , outputs  $\bar{w}$  such that  $\bar{\Omega}(\bar{w}, y, \hat{v}) = 1$ .  $\square$

In the above, we call an algorithm “efficient” if its time complexity is independent of the cardinality of the accumulated element set  $Y$ . Security of DUA is defined as follows. Capabilities of an adversary is defined through queries to oracle  $O_D$  which models a working DUA.  $O_D$  is initialized with the tuple  $(f, g, \mathcal{Y}_f, u, t_f)$  and maintains a list of elements  $Y$ , which is initially empty.  $O_D$  responds to two types of queries, namely “add  $y$ ” and “delete  $y$ .” It responds to an “add  $y$ ” query by adding  $y$  to the set  $Y$ , modifying the accumulator value  $v$  using algorithm  $D_1$  and sending back the updated accumulator value  $\hat{v}$ . It responds to a “delete  $y$ ” query by deleting it from set  $Y$ , modifying the accumulator value  $v$  using algorithm  $D_1$  and sending back the updated accumulator value  $\hat{v}$ . In the end,  $O_D$  outputs the current set  $Y$  and accumulator value  $v$ . The following is the definition of secure DUA.

**Definition 4 (Security of Dynamic Universal Accumulators (DUAs))** An universal accumulator is secure if, for any PPT algorithm  $\mathcal{A}$ ,  $P_3$  and  $P_4$  are negligible in  $\lambda$ , where:

$$\begin{cases} P_3 = \Pr \left[ \begin{array}{l} (f, g, \mathcal{Y}_f, u, t_f) \leftarrow \text{Gen}(1^\lambda); (y, w, Y) \leftarrow \mathcal{A}^{O_D(f, g, \mathcal{Y}_f, t_f)}(g \circ f, g, \mathcal{Y}_f) : \\ Y \subset \mathcal{Y}'_f \wedge y \in \mathcal{Y}'_f \setminus Y \wedge v = g \circ f(u, Y) \wedge \Omega(w, y, v) = 1 \end{array} \right] \\ P_4 = \Pr \left[ \begin{array}{l} (f, g, \mathcal{Y}_f, u, t_f) \leftarrow \text{Gen}(1^\lambda); (y, \bar{w}, Y) \leftarrow \mathcal{A}^{O_D(f, g, \mathcal{Y}_f, t_f)}(g \circ f, g, \mathcal{Y}_f) : \\ Y \subset \mathcal{Y}'_f \wedge y \in Y \wedge v = g \circ f(u, Y) \wedge \bar{\Omega}(\bar{w}, y, v) = 1 \end{array} \right] \end{cases}$$

$\square$

We state the following theorem. Its proof can be found in Appendix A.

**Theorem 1** A DUA is secure if the underlying UA is secure.  $\square$

Finally, a DUA-DDH is a DUA such that there exists a cyclic group  $\mathbb{G} \subset \mathcal{Y}_f$  in which the DDH assumption holds.

## 3.2 Constructions

We construct our DUA-DDH in stages. We first give a construction of UA for DDH groups. We then adds the necessary algorithms for enabling dynamism.

### 3.2.1 Our UA construction

This construction can be thought as the extension of Nguyen’s accumulator to support *universality*. Our computation of non-membership witnesses involves operations on polynomials over finite fields.

- **Generation** Let  $\lambda$  be a security parameter. Let  $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$  be a bilinear pairing such that  $|\mathbb{G}_1| = |\mathbb{G}_2| = p$  for some  $\lambda$ -bit prime  $p$ . Let  $g_0$  be a generator of  $\mathbb{G}_1$  and  $\mathbb{G}_q = \langle h \rangle$  be a cyclic group of prime order  $q$  such that  $\mathbb{G}_q \subset \mathbb{Z}_p^*$ .<sup>4</sup> The generation algorithm **Gen** randomly chooses  $\alpha \in_R \mathbb{Z}_p^*$ . For simplicity, we always take the initial element  $u = 1$ , the identity element in  $\mathbb{Z}_p^*$ . The function **f** is defined as  $f : \mathbb{Z}_p^* \times \mathbb{Z}_p^* \rightarrow \mathbb{Z}_p^*$  such that  $f : u, y \mapsto u(y + \alpha)$ . The function **g** is defined as  $g : \mathbb{Z}_p^* \times \mathbb{G}_1$  such that  $g : y \mapsto g_0^y$ . The domain  $\mathcal{Y}_f$  of accumulatable elements is  $\mathbb{G}_q$ .<sup>5</sup> The auxiliary information  $t_f$  is  $\alpha$ .
- **Evaluation** Computing  $g \circ f(1, Y)$  efficiently is straightforward with the auxiliary information  $\alpha$ . In case one wishes to allow computation of  $g \circ f$  without  $\alpha$ , one can publish  $g_0^{\alpha^i}$  for  $i = 0$  to  $k$ , where  $k$  is the maximum number of elements to be accumulated. If we denote the polynomial  $\prod_{y \in Y} (y + \alpha) = \sum_{i=0}^{i=k} (u_i \alpha^i)$  of maximum degree  $k$  as  $v(\alpha)$ , one can efficiently compute  $g \circ f(1, Y)$  as  $g_0^{v(\alpha)} = \prod_{i=0}^{i=k} g_i^{u_i} \in \mathbb{G}_1$ , without the knowledge of  $\alpha$ .
- **Membership witnesses** The relation  $\Omega$  is defined as  $\Omega(w, y, v) = 1$  if and only if  $\hat{e}(w, g_0^y g_0^\alpha) = \hat{e}(v, g_0)$ . For a set of elements  $Y := \{y_1, \dots, y_k\} \in \mathbb{G}_q$ , a membership witness for the element  $y \in Y$  can be computed in either one of the following ways, depending on whether one knows the auxiliary information.
  - (With auxiliary information.) Compute the witness as  $w = [g_0^{\prod_{i=1}^k (y_i + \alpha)}]_{\alpha + y}^{\frac{1}{\alpha + y}}$ .
  - (Without auxiliary information.) Let  $w(\alpha)$  be the polynomial  $\prod_{i=1, i \neq j}^k (y_i + \alpha)$ . Expand  $w$  and write it as  $w(\alpha) = \sum_{i=0}^{i=k-1} (u_i \alpha^i)$ . Compute the witness as  $w = g_0^{w(\alpha)} = \prod_{i=0}^{i=k-1} g_i^{u_i} \in \mathbb{G}_1$ .
- **Non-membership witnesses** The relation  $\bar{\Omega}$  for non-membership witnesses is defined as  $\bar{\Omega}(\bar{w}, y, v) = 1$  if and only if  $\bar{w} = (c, d)$  and  $\hat{e}(c, g_0^y g_0^\alpha) \hat{e}(g_0, g_0)^d = \hat{e}(v, g_0)$ . For a set of elements  $Y := \{y_1, \dots, y_k\} \in \mathbb{G}_q$ , a non-membership witness for  $\tilde{y} \notin Y$  can be computed in either one of the following ways, depending on whether one knows the auxiliary information:
  - (With auxiliary information.) Compute  $\bar{w} = (c, d)$  according to  $d = \prod_{i=1}^k (y_i + \alpha) \bmod (\alpha + \tilde{y}) \in \mathbb{Z}_p$  and  $c = g_0^{\frac{\prod_{i=1}^k (y_i + \alpha) - d}{\tilde{y} + \alpha}} \in \mathbb{G}_1$ .
  - (Without auxiliary information.) Denote the polynomial  $v(\alpha)$  as  $\prod_{i=1}^k (y_i + \alpha)$ . Compute a polynomial division of  $v(\alpha)$  by  $(\alpha + \tilde{y})$ . Since  $(\alpha + \tilde{y})$  is a degree one polynomial and  $\tilde{y} \neq y_i$  for all  $i$ , there exists a degree  $k - 1$  polynomial  $c(\alpha)$  and a constant  $d$  such that  $v(\alpha) = c(\alpha)(\alpha + \tilde{y}) + d$ . Expand  $c$  and write it as  $c(\alpha) = \sum_{i=0}^{i=k-1} (u_i \alpha^i)$ . Compute  $c = g_0^{c(\alpha)} = \prod_{i=0}^{i=k-1} g_i^{u_i} \in \mathbb{G}_1$ . The non-membership witness of  $\tilde{y}$  is  $\bar{w} = (c, d)$ .

<sup>4</sup>If  $p = 2q + 1$ , one can choose a random element in  $h \in_R \mathbb{Z}_p^*$  with order  $q$  and set  $\mathbb{G}_q = \langle h \rangle$ .

<sup>5</sup>Formally, it is  $\mathbb{G}_q \setminus \{-\alpha\}$ .

The theorem below states the security of our UA. Its proof can be found in Appendix A.

**Theorem 2 (Security of our UA construction)** Under the  $k$ -SDH assumption in  $\mathbb{G}_1$ , the above construction is a secure universal accumulator.  $\square$

### 3.2.2 Our DUA-DDH construction

We present our construction of DUA-DDH by adding the various dynamism algorithms  $D_1, D_2, D_3$  to our UA construction above. Due to Theorem 1 and 2, our construction is secure under the  $k$ -SDH assumption.

- **Update of accumulator (algorithm  $D_1$ )** Adding an element  $\hat{y}$  to the accumulator value  $v$  can be done by computing  $\hat{v} = v^{\hat{y}+\alpha}$ . Similarly, deleting an element  $\hat{y}$  in the accumulator  $v$  can be done by computing  $\hat{v} = v^{\frac{1}{\hat{y}+\alpha}}$ . Both cases require the auxiliary information  $\alpha$ .
- **Update of membership witnesses (algorithm  $D_2$ )** Let  $w$  be the original membership witness of  $y$  w.r.t the accumulator value  $v$ . Let  $\hat{v}$  and  $\hat{y}$  be the new accumulator value and the element added (resp. deleted) respectively. Suppose  $\hat{y}$  has been added, the new membership witness  $\hat{w}$  for  $y$  can be computed as  $v w^{\hat{y}-y}$ . Suppose  $\hat{y} \neq y$  has been deleted, the new non-membership witness  $\hat{w}$  for  $y$  can be computed as  $w^{\frac{1}{\hat{y}-y}} \hat{v}^{\frac{1}{y-\hat{y}}}$ .
- **Update of non-membership witnesses (algorithm  $D_3$ )** Let  $c, d$  be the original non-membership witness of  $y$  w.r.t. accumulator value  $v$ . Let  $\hat{v}$  and  $\hat{y}$  be the new accumulator value and the element added (resp. deleted) respectively.
  - (*Addition.*) Suppose  $\hat{y} \neq y$  has been added, the new non-membership witness  $\hat{c}, \hat{d}$  of  $y$  can be computed as  $\hat{c} = v c^{\hat{y}-y} \in \mathbb{G}_1$  and  $\hat{d} = d(\hat{y} - y) \in \mathbb{Z}_p^*$ . This can be verified as follows:

$$\begin{aligned} \hat{v} &= v^{\alpha+\hat{y}} = v^{(\alpha+y)+(\hat{y}-y)} = v^{\alpha+y} v^{\hat{y}-y} = v^{\alpha+y} (c^{\alpha+y} g_0^d)^{\hat{y}-y} \\ &= [v c^{\hat{y}-y}]^{\alpha+y} g_0^{d(\hat{y}-y)} = \hat{c}^{\alpha+y} g_0^{\hat{d}} \end{aligned}$$

- (*Deletion.*) Suppose  $\hat{y}$  has been deleted, the new non-membership witness  $\hat{c}, \hat{d}$  of  $y$  can be computed as  $\hat{c} = (c\hat{v}^{-1})^{\frac{1}{\hat{y}-y}} \in \mathbb{G}_1$  and  $\hat{d} = \frac{d}{\hat{y}-y} \in \mathbb{Z}_p^*$ . Indeed,

$$\begin{aligned} \hat{v} &= \hat{v}^{\frac{(\alpha+\hat{y})-(\alpha+y)}{\hat{y}-y}} = v^{\frac{1}{\hat{y}-y}} \hat{v}^{\frac{\alpha+y}{y-\hat{y}}} = [c^{\alpha+y} g_0^d]^{\frac{1}{\hat{y}-y}} \hat{v}^{\frac{\alpha+y}{y-\hat{y}}} \\ &= [(c\hat{v}^{-1})^{\alpha+y} g_0^d]^{\frac{1}{\hat{y}-y}} = [(c\hat{v}^{-1})^{\frac{1}{\hat{y}-y}}]^{\alpha+y} g_0^{\frac{d}{\hat{y}-y}} = \hat{c}^{\alpha+y} g_0^{\hat{d}} \end{aligned}$$

## 4 Zero-Knowledge Protocols for Our DUA-DDH

We present several efficient zero-knowledge protocols for our DUA-DDH construction. In the presentation, we give priority to clarity over efficiency; the protocols may be optimized for better performance.

Let  $\mathbb{G}_1 = \langle \mathbf{g} \rangle$  and  $\mathbb{G}_q = \langle \mathbf{h} \rangle$  be cyclic groups of prime order  $p$  and  $q$  respectively, such that  $\mathbb{G}_q \subset \mathbb{Z}_p^*$  is the domain of our DUA-DDH construction. Let  $\mathbf{g}_0, \mathbf{g}_1$  and  $\mathbf{h}_0, \mathbf{h}_1, \mathbf{h}_2$  be independent generators of  $\mathbb{G}_1$  and  $\mathbb{G}_q$  respectively. Let  $y = \mathbf{h}_0^x \in \mathbb{G}_q$  and let  $\mathfrak{C} = \mathbf{g}_0^y \mathbf{g}_1^r \in \mathbb{G}_1$  be the commitment of  $y$  using random number  $r$ . Let  $v$  be an accumulator value.

### 4.1 Proof of knowledge of the discrete logarithm of a committed element

This protocol is the main building block of the protocols used in our DUA-DDH construction. We call it  $\text{PK}_1$ . Let  $\mathfrak{D} = \mathbf{h}_1^x \mathbf{h}_2^s \in \mathbb{G}_q$  be the commitment of  $x$  using some random number  $s$ . The goal of  $\text{PK}_1$  is to prove the knowledge of  $x$  and  $y$  such that  $y = \mathbf{h}_0^x$  in zero-knowledge, thus without revealing, e.g.,  $x$  or  $y$ . In other words, we have:

$$\text{PK}_1 \left\{ (y, r, x, s) : \mathfrak{C} = \mathbf{g}_0^y \mathbf{g}_1^r \wedge \mathfrak{D} = \mathbf{h}_1^x \mathbf{h}_2^s \wedge y = \mathbf{h}_0^x \right\}$$

The protocol can be used with the common discrete logarithm relationship proofs [Cam98] to demonstrate relationships of discrete logarithms in  $\mathbb{G}_1$  or  $\mathbb{G}_q$ . Instantiation of  $\text{PK}_1$  makes use of the zero-knowledge proof-of-knowledge of double discrete logarithms [CS97], as we now describe. Let  $\lambda_k$  be a security parameter that determines the cheating probability of the protocol. (The cheating probability is  $2^{-\lambda_k}$ , we hence suggest  $\lambda_k = 80$ .)  $\text{PK}_1$  consists of  $\text{PK}_{1A}$  and  $\text{PK}_{1B}$  as follows.

$$\text{PK}_1 \left\{ \begin{array}{l} \text{PK}_{1A} \left\{ (y, r) : \mathfrak{C} = \mathbf{g}_0^y \mathbf{g}_1^r \right\} \\ \text{PK}_{1B} \left\{ (x, r, s) : \mathfrak{C} = \mathbf{g}_0^{\mathbf{h}_0^x} \mathbf{g}_1^r \wedge \mathfrak{D} = \mathbf{h}_1^x \mathbf{h}_2^s \right\} \end{array} \right\}$$

Instantiating  $\text{PK}_{1A}$  is straightforward. Below we only show how to instantiate  $\text{PK}_{1B}$ .

(Commitment.) For  $i = 1$  to  $\lambda_k$ , the prover randomly generates  $\rho_{x,i}, \rho_{s,i} \in_R \mathbb{Z}_q$  and  $\rho_{r,i} \in_R \mathbb{Z}_p$ , computes  $T_{1,i} = \mathbf{g}_0^{\rho_{x,i}} \mathbf{g}_1^{\rho_{r,i}} \in \mathbb{G}_1$  and  $T_{2,i} = \mathbf{h}_1^{\rho_{x,i}} \mathbf{h}_2^{\rho_{s,i}} \in \mathbb{G}_q$ , and sends  $T_{1,i}, T_{2,i}$  to the verifier.

(Challenge.) The verifier randomly generates a  $\lambda_k$ -bit challenge  $m$  and sends it to the prover.

(Response.) Denote by  $m[i]$  the  $i$ -th bit of  $m$ , starting from  $i = 1$ . For  $i = 1$  to  $\lambda_k$ , the prover computes  $z_{x,i} = \rho_{x,i} - m[i]x \in \mathbb{Z}_q$ ,  $z_{s,i} = \rho_{s,i} - m[i]s \in \mathbb{Z}_q$  and  $z_{r,i} = \rho_{r,i} - m[i]\mathbf{h}_0^x r \in \mathbb{Z}_p$ . She sends  $(z_{x,i}, z_{s,i}, z_{r,i})_{i=1}^{\lambda_k}$  to the verifier.

(Verify.) The verifier outputs 1 if the following holds for all  $i = 1$  to  $\lambda_k$ . He outputs 0 otherwise.

$$T_{2,i} \stackrel{?}{=} \mathfrak{D}^{m[i]} \mathfrak{h}_1^{z_{x,i}} \mathfrak{h}_2^{z_{s,i}} \quad \text{and} \quad T_{1,i} \stackrel{?}{=} \begin{cases} \mathfrak{g}_0^{z_{x,i}} \mathfrak{g}_1^{z_{r,i}}, & \text{if } m[i] = 0, \\ \mathfrak{c} \mathfrak{h}_0^{z_{x,i}} \mathfrak{g}_1^{z_{r,i}}, & \text{otherwise.} \end{cases}$$

It is straightforward to show that  $\text{PK}_1$  is Honest-Verifier Zero-Knowledge. It can be converted into a 4-round perfect zero-knowledge protocol using the technique due to Cramer et al. [CDM00] or 3-move concurrent zero-knowledge protocol in the auxiliary string model based on trapdoor commitment schemes [Dam00]. Note that the prover does not need to explicitly prove that the  $r$  in  $\text{PK}_{1A}$  and  $\text{PK}_{1B}$  are the same; they are bounded to be the same under the discrete logarithm assumption.

## 4.2 Proof of knowledge of a committed element in an accumulator value

Suppose  $y$  is in the accumulator value  $v$ . That is, there exists witness  $w$  such that  $\Omega(w, y, v) = 1$ . The following protocol demonstrates that the element  $y$ , committed as  $\mathfrak{C}$ , is in the accumulator value  $v$ .

$$\text{PK}_2 \left\{ (w, y, r) : \hat{e}(w, g_0^y g_0^\alpha) = \hat{e}(v, g_0) \wedge \mathfrak{C} = \mathfrak{g}_0^y \mathfrak{g}_1^r \right\}$$

$\text{PK}_2$  can be instantiated using the standard proof-of-knowledge of an SDH-tuple [BBS04, ASM06].

Combining  $\text{PK}_1$  and  $\text{PK}_2$ , we have a protocol, denoted as  $\text{PK}_3$ , that proves the knowledge of the discrete logarithm of an element in an accumulator value:

$$\text{PK}_3 \left\{ (w, y, x) : \hat{e}(w, g_0^y g_0^\alpha) = \hat{e}(v, g_0) \wedge y = \mathfrak{h}_0^x \right\}$$

## 4.3 Proof of knowledge of a committed element not in an accumulator value

Suppose  $y$  is *not* in the accumulator value  $v$ . Then there exists witness  $\bar{w} = (c, d)$  such that  $d \neq 0$  and  $\bar{\Omega}(\bar{w}, y, v) = 1$ . The following protocol demonstrates that the element  $y$ , committed as  $\mathfrak{C}$ , is *not* in the accumulator value  $v$ .

$$\text{PK}_4 \left\{ (c, d, y, r) : \hat{e}(c, g_0^y g_0^\alpha) = \hat{e}(v, g_0) \hat{e}(g_0, g_0)^d \wedge d \neq 0 \wedge \mathfrak{C} = \mathfrak{g}_0^y \mathfrak{g}_1^r \right\}$$

$\text{PK}_4$  can be instantiated using standard techniques. For completeness, its instantiation is shown in Appendix B.

Combining  $\text{PK}_1$  and  $\text{PK}_4$ , we have a protocol, denoted as  $\text{PK}_5$ , that proves the knowledge of the discrete logarithm of an element not in an accumulator value:

$$\text{PK}_5 \left\{ (c, d, y, x) : \hat{e}(c, g_0^y g_0^\alpha) \hat{e}(g_0, g_0)^d = \hat{e}(v, g_0) \wedge d \neq 0 \wedge y = \mathfrak{h}_0^x \right\}$$

## 5 Our Attribute-Based Anonymous Credential System

We first formally define the syntax and security model for ABACS. Next, we present our construction, followed by security proof.

### 5.1 Syntax

An ABACS is a tuple of six algorithms/protocols (Attribute-CAGen, UserGen, Nym, CertifyAttribute, RevokeAttribute, Authentication), between three parties, namely, an ACA  $O$ , a User  $U$  and any Verifier  $V$ .

**Attribute-CAGen** This is the key generation algorithm of an Attribute CA  $O_j$ , who is responsible for certifying attribute  $A_j$ .

**UserGen** This is the key generation algorithm of a user  $U_i$ .

**Nym** This is an interactive protocol between  $U_i$  and  $O_j$  to generate a pseudonym  $y_{i,j}$ . A pseudonym is a piece of bit-string that  $O_j$  recognizes  $U_i$ .

**CertifyAttribute** This protocol allows  $O_j$  to certify attribute  $A_j$  for  $U_i$ , whom he recognized by pseudonym  $y_{i,j}$ . We say  $U_i$  is a members of  $O_j$  or  $U_i$  possesses attribute  $A_j$  upon successful completion of the protocol.

**RevokeAttribute** This algorithm allows  $O_j$  to take away the attribute  $A_j$  issued to  $U_i$  whom he recognized by pseudonym  $y_{i,j}$ .

**Authentication** This interactive protocol allows  $U_i$  to demonstrate to any verifier  $V$  that he is in possession of a set of attributes satisfying certain policy. A policy  $\mathcal{ST}$  (in DNF) is of the following form:  $\mathcal{ST} = \bigvee (\mathcal{ST}_k)_{k=1}^{\ell}$  such that  $\mathcal{ST}_k$  is the conjunction of any number of the following statements.

1.  $U_i$  possesses attribute  $A_j$
2.  $U_i$  is the owner of a pseudonym  $y$  with  $O_{j^*}$ .
3. (*Negation of 1.*)  $U_i$  does not possess attribute  $A_j$
4. (*Negation of 2.*)  $U_i$  is not the owner of a pseudonym with  $O_{j^*}$ .

### 5.2 Security model

We define a security model to capture the security requirements using a simulation-based approach, in the sense of [CL01]. Firstly we define an ABACS that relies on a trusted party  $\mathcal{T}$  as an intermediary. We assume communication between  $\mathcal{T}$  and any other parties is secure. This is sometimes referred to as an ideal-world specification of an ABACS. Next we define what it means for a cryptographic authentication system to conform to an ideal-world specification.

### 5.2.1 An informal description

We present an informal description of the requirements of an ABACS first.

User authenticity The verifier can specify any access policies, and a user can successfully authenticate *if and only if* the set of attributes he possesses satisfy the policy. In particular, *user authenticity* must be *collusion-resistant*: no collusion of users who do not individually satisfy a policy can successfully authenticate w.r.t. that policy.

User anonymity Authentication of users with respect to a policy is anonymous and unlinkable among the set of all users who satisfy the policy. This is in fact the minimum amount of information the user must leak to the server for a secure access control.

Attribute-CA autonomy ACAs operate autonomously: an ACA can certify (and possibly later revoke) an attribute without needing to interact with other ACAs.

Attribute Dynamism New ACAs can be introduced without affecting existing users. Each ACA can certify (and revoke) users' attribute independently. Existing users do not need to contact the ACA to reflect the changes.

Attribute privacy Collusion of ACAs together cannot find out what attributes a user possesses.

Support for attribute negation Users can authenticate their non-possession of an attribute *without* having to contact (and hence be certified by) the respective ACA in advance.

### 5.2.2 Formal model

The formal model is given below.

**Ideal-world ABACS system** We describe the ideal-world ABACS system (IAS) that relies on a trusted party  $\mathcal{T}$  as an intermediary. An IAS consists of a trusted party  $\mathcal{T}$  through which all transactions are carried out and a set of honest ideal players. In IAS, the ideal players are user  $U$ , verifier  $V$  and ACA  $O$ .

Initialization: The system is initialized when every ACA  $O_j$  creates a list  $\mathcal{L}_j$ .  $\mathcal{L}_j$  represents the member list of  $O_j$ , which is empty at this stage. We say user is in possession of attribute  $A_j$  if and only if he is in  $\mathcal{L}_j$ .

Ideal communication: All communications are routed through  $\mathcal{T}$ . The sender can request  $\mathcal{T}$  not to reveal his identity to the recipient if he wishes to be anonymous. The sender may also request to establish a session between him and the recipient. In particular, we assume some kind of authentication is implemented between  $\mathcal{T}$  and each player. For instance, user  $U_i$  cannot contact  $\mathcal{T}$ , claiming himself to be another user  $U_j$ .

Events in the system: Each transaction between players is an event in the system. Events can be triggered through external processes or controlled by an adversary. An external process can trigger some particular event between a particular user and an ACA; or may trigger a set of events; or may cause some probability distribution on the events.

Output of the players: In the end of the system’s lifetime, each user outputs a list of the transactions he participated in and the transaction outcomes.

Transactions: The system supports the following transactions.

- $\text{Nym}(U_i, O_j)$ : This protocol is a session between user  $U_i$  and ACA  $O_j$ .  $U_i$  first contacts  $\mathcal{T}$ , requesting to establish a pseudonym between himself and  $O_j$ . If  $U_i$  and  $O_j$  have already established a pseudonym,  $\mathcal{T}$  simply replies with the previous one. Otherwise,  $\mathcal{T}$  picks a pseudonym  $y_{i,j}$  for them and informs both  $U_i$  and  $O_j$  the value  $y_{i,j}$ .  $\mathcal{T}$  stores  $y_{i,j}$  as the pseudonym of user  $U_i$  with  $O_j$ .
- $\text{CertifyAttribute}(U_i, O_j)$ : This protocol is a session between  $U_i$  and  $O_j$ .  $U_i$  first contacts  $\mathcal{T}$  with  $y_{i,j}$ . If  $y_{i,j}$  is not the pseudonym of user  $U_i$  with  $O_j$ ,  $\mathcal{T}$  replies “Fail”. Otherwise  $\mathcal{T}$  contacts  $O_j$  with a certification request for pseudonym  $y_{i,j}$ . If  $O_j$  accepts the request,  $\mathcal{T}$  notifies  $U_i$  that his request is granted.
- $\text{RevokeAttribute}(O_j, U_i)$ : This algorithm allows  $O_j$  to revoke the attribute of  $U_i$ .  $O_j$  removes  $y_{i,j}$  from  $\mathcal{L}_j$  and notify  $\mathcal{T}$  the updated  $\mathcal{L}_j$ . This ideal functionality is defined to give the adversary the extra power of influencing an honest  $O_j$  to revoke attributes of some of its users.
- $\text{Authentication}(U_i, V, \mathcal{ST})$ : This protocol is a session between  $U_i$  and  $V$  regarding a policy  $\mathcal{ST}$  which is the conjunction and disjunction of any number of the following, in disjunctive normal form:
  1.  $U_i$  possesses attribute  $A_j$  (and its negation).
  2.  $U_i$  is the owner of pseudonym  $y$  with  $O_j$  (and its negation).

For each  $O_j$  that appears in  $\mathcal{ST}$  (concerning attribute  $A_j$ ),  $\mathcal{T}$  contacts  $O_j$  and obtains  $\mathcal{L}_j$ .  $\mathcal{T}$  forwards the set of lists  $\{\mathcal{L}_j\}$  to  $U_i$ . Upon receiving the list,  $U_i$  decides if he wishes to continue the protocol and if that is the case,  $\mathcal{T}$  checks from the set of  $\{\mathcal{L}_j\}$  and his list of pseudonym  $\{y_{i,j}\}$  and decides if  $U_i$  fulfills the policy  $\mathcal{ST}$ .  $\mathcal{T}$  contacts  $V$  with  $\mathcal{ST}$ , along with the set of lists  $\{\mathcal{L}_j\}$ , together with a bit indicating whether the underlying user fulfills  $\mathcal{ST}$  or not.  $V$  sends his response to  $\mathcal{T}$ .  $\mathcal{T}$  forwards the response to  $U_i$ .

Intuitively, this ideal-world system captures the security requirements of an ABACS.

**Cryptographic ABACS** A cryptographic ABACS system (CAS) consists of a set of honest cryptographic players. In CAS, these players are user  $U$ , verifier  $V$  and ACA  $O$ .

**The ideal-world (resp. real-world) adversary** The ideal-world (resp. real-world) adversary is a PPT that gets control over the corrupted parties in the ideal world (resp. real-world). He receives the number of honest users and organizations and public information of the system as input. The adversary can also trigger an event as described above.

**Definition of secure CAS** A CAS is secure if it conforms to an ideal-world specification. Informally speaking, CAS is said to be conformed to an ideal-world specification if there exists a simulator  $\mathcal{S}$  such that for any real-world adversary  $\mathcal{A}$ ,  $\mathcal{S}$ , with black-box access to  $\mathcal{A}$ , acts as an ideal-world adversary so that the output of all honest players in IAS is computationally indistinguishable to the output of all honest players in CAS.  $\mathcal{S}$  represents adversary-controlled parties in IAS (ideal-world), and honest parties in CAS (real-world).  $\mathcal{S}$  translates a real-world adversary into an ideal-world adversary. Existence of such simulator implies that for any PPT algorithm (real-world adversary) in CAS, there exists a PPT algorithm (ideal-world adversary) in the ideal-world such that the outputs of the corresponding honest parties in the two worlds are the same. Since IAS is secure (any PPT in IAS cannot do anything that breach the security requirements), CAS is secure. Specifically, we have the following definition.

**Definition 5** *Let IAS be the ideal-world ABACS. Let CAS be a cryptographic ABACS. For a security parameter  $\lambda$ , let the number of players in the system be polynomial in  $\lambda$ . By  $IAS(1^\lambda, E)$  (resp.,  $CAS(1^\lambda, E)$ ) we denote an ABACS with security parameter  $\lambda$  and event scheduler  $E$  for the events that have taken place in the system. Let  $\mathcal{A}$  be the real-world adversary. Since  $E$  schedules the event according to  $\mathcal{A}$ 's wishes, we denote it as  $E^\mathcal{A}$ . By  $Z_i(1^\lambda)$  we denote the output of (honest) party  $i$  in the system. If  $\{Z_1(1^\lambda), \dots, Z_\ell(1^\lambda)\}$  is a list of players' outputs, then we denote these players' output by  $\{Z_1(1^\lambda), \dots, Z_\ell(1^\lambda)\}^{AS(1^\lambda, E)}$  when all of them together exist within an ABACS. CAS is secure if there exists a simulator  $\mathcal{S}$  such that the following holds, for all PPT  $\mathcal{A}$  and for all sufficiently large  $\lambda$ :*

1. *In IAS,  $\mathcal{S}$  controls the ideal-world players corresponding to the real-world players controlled by  $\mathcal{A}$ .*
2. *For all event schedulers  $E^\mathcal{A}$*

$$\{\{Z_i(1^\lambda)\}_{i=1}^\ell, \mathcal{A}(1^\lambda)\}^{CAS(1^\lambda, E)} \stackrel{?}{\approx} \{\{Z_i(1^\lambda)\}_{i=1}^\ell, \mathcal{S}^\mathcal{A}(1^\lambda)\}^{IAS(1^\lambda, E)}$$

where  $\mathcal{S}$  is given black-box access to  $\mathcal{A}$  and  $D_1(1^\lambda) \stackrel{?}{\approx} D_2(1^\lambda)$  denotes computational indistinguishability of the two distributions  $D_1$  and  $D_2$ .

### 5.3 Construction

Description of our construction of ABACS from our DUA is given below, followed by some discussion on complexity issues.

**Common Parameter.** We assume  $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$  such that  $|\mathbb{G}_1| = |\mathbb{G}_2| = p$  and  $\mathbb{G}_q \subset \mathbb{Z}_p^*$  with  $|\mathbb{G}_q| = q$  are system-wide parameter.

**Attribute-CAGen.** Each  $O_j$  maintains its own DUA ( $g_j \circ f_j$  working in the bilinear group pair  $\mathbb{G}_1, \mathbb{G}_2$ ) and a random element  $h_j \in \mathbb{G}_q$ . To ensure  $h_j$  is chosen randomly, we sets  $h_j = H(A_j) \in \mathbb{G}_q$ , where  $A_j$  represents the attribute certified by ACA  $O_j$ ,

for some cryptographic hash function  $H$ . Define  $h_j : x \mapsto h_j^x$ .  $O_j$  also maintains two public lists,  $\mathcal{L}_j$  and  $\mathcal{L}'_j$  which are initialized to empty list.

**UserGen.** User  $U_i$  chooses his secret key  $x_i \in_R \mathbb{Z}_q$ .

**Nym.** User  $U_i$  sends  $y_{i,j} = h_j(x_i)$  to  $O_j$  as his pseudonym. Upon successful completion of the protocol, we say  $U_i$  is the owner of the pseudonym  $y_{i,j}$ .

**CertifyAttribute.** ACA  $O_j$  certifies attribute  $A_j$  to  $U_i$  by setting  $\mathcal{L}_j := \mathcal{L}_j \cup \{y_{i,j}\}$ . For efficiency reason,  $O_j$  returns to  $U_i$  the membership witness  $w_{i,j}$  such that  $y_{i,j}$  is in the accumulator  $v_j = g_j \circ f_j(1, \mathcal{L}_j)$ <sup>6</sup>  $O_j$  also appends the entry  $(v_j, y_{i,j}, \text{ADD})$  to list  $\mathcal{L}'_j$ .

**RevokeAttribute.** ACA  $O_j$  sets  $\mathcal{L}_j := \mathcal{L}_j \setminus \{y_{i,j}\}$ .  $O_j$  computes  $v_j = g_j \circ f_j(1, \mathcal{L}_j)$  and appends the entry  $(v_j, y_{i,j}, \text{REMOVE})$  to list  $\mathcal{L}'_j$ . Users use the dynamism algorithms  $D_2$  (resp.  $D_3$ ) of our DUA to update its membership witness (resp. nonmembership witness) with the list  $\mathcal{L}'_j$ .

**Authentication.** Consider a policy  $\mathcal{ST}$  (in DNF) of the following form:  $\mathcal{ST} = \bigvee (\mathcal{ST}_k)_{k=1}^\ell$  such that  $\mathcal{ST}_k$  is the conjunction of any number of the following statements.

1.  $U_i$  possesses attribute  $A_j$
2.  $U_i$  is the owner of a pseudonym  $y$  with  $O_{j^*}$ .
3. (*Negation of 1*).  $U_i$  does not possess attribute  $A_j$
4. (*Negation of 2*).  $U_i$  is not the owner of a pseudonym with  $O_{j^*}$ .

Each statement above has a corresponding zero-knowledge proof-of-knowledge statement.

1.  $h_j(x_i)$  is in  $\mathcal{L}_j$ . This can be done using  $\text{PK}_2$  described in Section 4.
2.  $y = h_{j^*}(x_i)$ . This can be done using  $\text{PK}_4\{(x_i) : y = H(A_{j^*})^{x_i}\}$  which is just the standard zero-knowledge proof-of-knowledge of discrete logarithm.
3. (*Negation of 1*).  $h_j(x_i)$  is not in  $\mathcal{L}_j$ . This can be done using  $\text{PK}_3$  described in Section 4.
4. (*Negation of 2*).  $y \neq h_{j^*}(x_i)$ . This can be done using  $\text{PK}_5\{(x_i) : y \neq H(A_{j^*})^{x_i}\}$  which is the zero-knowledge proof-of-knowledge of inequality of discrete logarithm [CS03].

Thus, to assert  $\mathcal{ST}$ , one just need to conduct a zero-knowledge proof-of-knowledge by the disjunction of conjunction of  $\text{PK}_2$ ,  $\text{PK}_3$ ,  $\text{PK}_4$ ,  $\text{PK}_5$ , together with the proof that all  $x_i$  used in those proofs are equal, which can be done by the employing technique from [CDS94].

---

<sup>6</sup>This is not necessary since the user can compute the witness by himself from  $\mathcal{L}_j$  but  $O_j$  can compute the value more efficiently with the auxiliary information.

**Complexity issues** The most expensive operation is the use of ZKPoK of double discrete logarithm (that is,  $\text{PK}_1$ ), which has time and space complexities of  $O(\lambda_k)$ , where  $\lambda_k$  is the security parameter regarding the cheating probability of the protocol<sup>7</sup>. Instantiations of other protocols are of constant time and space complexities. For a policy  $\mathcal{ST} = \bigvee (\mathcal{ST}_k)_{k=1}^\ell$ , the total complexities is  $O(\ell\lambda_k)$  where  $O(\ell\lambda_k)$  is the complexities of  $\mathcal{ST}_k$  and is linearly dependent to the number of attributes involved in  $\mathcal{ST}_k$  times  $\lambda_k$ . Indeed, we can optimize that protocol so that at most one ZKPoK of double discrete logarithm (that is,  $\text{PK}_1$ ) is needed in each of the  $\mathcal{ST}_k$ . Furthermore, due to the nature of [CDS94], our system supports threshold version of policy efficiently. That is, for a threshold value  $\tilde{k} \leq \ell$ , our system supports policy of the form  $\mathcal{ST} = \bigvee_{|\mathcal{K}|=\tilde{k}, \mathcal{K} \subset [1, \dots, \ell]} \left( \bigwedge_{k \in \mathcal{K}} (\mathcal{ST}_k)_{k=1}^\ell \right)$  with the same complexities as  $\mathcal{ST} = \bigvee (\mathcal{ST}_k)_{k=1}^\ell$ .

**Security** If  $k$  is the maximum number of members for any ACA, we have the following theorem regarding the security of our ABACS.

**Theorem 3 (Security of our ABACS construction)** Under the  $k$ -SDH assumption in  $\mathbb{G}_1$ , the DDH assumption in  $\mathbb{G}_q$ , our ABACS is secure in the random oracle model.  $\square$

Its proof can be found in Appendix A.

## 6 Concluding Remarks

We have presented the first dynamic universal accumulator construction for accumulating elements in DDH-hard groups and a number of useful zero-knowledge protocols for it. Using this accumulator, we have built the *Attribute-Based Anonymous Credential System*, which allows the verifier to authenticate anonymous users according to any access control policy expressible as formula of boolean user attributes in the DNF form. Our system features many practicality and scalability properties for a large-scale deployment of privacy-preserving access control in a heterogeneous and decentralized environment.

We end the paper with two research questions that we believe to be worth exploring in the future. The first one is how one can construct ABACS that also efficiently supports numeric attributes. (While one could certainly encode a numerical attribute by a bunch of boolean attributes, that wouldn't be very efficient.) The second question is how one can construct ABACS that avoids the need to prove double discrete logarithms, and hence achieves better efficiency.

---

<sup>7</sup>Since the cheating probability is  $2^{-\lambda_k}$ , we suggest  $\lambda_k$  to be at least 80.

## References

- [ASM06] Man Ho Au, Willy Susilo, and Yi Mu. Constant-Size Dynamic  $k$ -TAA. In *SCN*, volume 4116, pages 111–125, 2006.
- [BB04] Dan Boneh and Xavier Boyen. Short Signatures Without Random Oracles. In *EUROCRYPT*, volume 3027 of *LNCS*, pages 56–73, 2004.
- [BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short Group Signatures. In *CRYPTO*, volume 3152 of *LNCS*, pages 41–55, 2004.
- [BCKL08] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. P-signatures and Noninteractive Anonymous Credentials. In *TCC*, pages 356–374, 2008.
- [BdM93] Josh Cohen Benaloh and Michael de Mare. One-Way Accumulators: A Decentralized Alternative to Digital Signatures (Extended Abstract). In *EUROCRYPT*, volume 765 of *LNCS*, pages 274–285, 1993.
- [BP97] Niko Barić and Birgit Pfitzmann. Collision-Free Accumulators and Fail-Stop Signature Schemes Without Trees. In *EUROCRYPT*, volume 1233 of *LNCS*, pages 480–494, 1997.
- [BSW07] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334. IEEE Computer Society, 2007.
- [Cam98] Jan Camenisch. Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem. *PhD Thesis, ETH Zürich, 1998. Diss. ETH No. 12520, Hartung Gorre Verlag, Konstanz., 1998.*
- [CDM00] Ronald Cramer, Ivan Damgård, and Philip D. MacKenzie. Efficient Zero-Knowledge Proofs of Knowledge Without Intractability Assumptions. In *Public Key Cryptography*, volume 1751 of *LNCS*, pages 354–373, 2000.
- [CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In *CRYPTO*, volume 839 of *LNCS*, pages 174–187, 1994.
- [CL01] Jan Camenisch and Anna Lysyanskaya. An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In *EUROCRYPT*, volume 2045 of *LNCS*, pages 93–118, 2001.
- [CL02a] Jan Camenisch and Anna Lysyanskaya. A Signature Scheme with Efficient Protocols. In *SCN*, volume 2576 of *LNCS*, pages 268–289, 2002.
- [CL02b] Jan Camenisch and Anna Lysyanskaya. Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In *CRYPTO*, volume 2442 of *LNCS*, pages 61–76, 2002.

- [CL04] Jan Camenisch and Anna Lysyanskaya. Signature Schemes and Anonymous Credentials from Bilinear Maps. In *CRYPTO*, volume 3152 of *LNCS*, pages 56–72, 2004.
- [CS97] Jan Camenisch and Markus Stadler. Efficient Group Signature Schemes for Large Groups (Extended Abstract). In *CRYPTO'97*, volume 1294 of *LNCS*, pages 410–424, 1997.
- [CS03] Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *CRYPTO*, volume 2729 of *LNCS*, pages 126–144, 2003.
- [Dam00] Ivan Damgård. Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. In *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 418–430, 2000.
- [FH02] S. Farrell and R. Housley. An Internet Attribute Certificate Profile for Authorization, 2002.
- [GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM J. Comput.*, 18(1):186–208, 1989.
- [LLX07] Jiangtao Li, Ninghui Li, and Rui Xue. Universal Accumulators with Efficient Nonmembership Proofs. In *ACNS*, volume 4521 of *LNCS*, pages 253–269, 2007.
- [Ngu05] Lan Nguyen. Accumulators from Bilinear Pairings and Applications. In *CT-RSA 2005*, volume 3376 of *LNCS*, pages 275–292, 2005.
- [Ped91] Torben P. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *CRYPTO*, volume 576 of *LNCS*, pages 129–140. Springer, 1991.

## A Security Proofs

### A.1 Proof of Theorem 1

**Proof 1** Let  $\mathcal{A}$  be an adversary that breaks the security definition in Definition 4 of an DUA, we show how to construct a simulator  $\mathcal{S}$  which breaks the security definition of the underlying UA.

Now we assume  $\mathcal{A}$  breaks the security property by outputting a membership witness (resp. nonmembership witness) for an element that is not inside (resp. inside) the accumulator. On input  $(f, g)$ ,  $\mathcal{S}$  gives the value  $g \circ f$  to  $\mathcal{A}$ .  $\mathcal{S}$  simulates the oracle  $O_D$  by computing the accumulation of the updated set. For instance, when  $\mathcal{A}$  sends add  $y$  query,  $\mathcal{S}$  inserts  $y$  to the set  $Y$ , and computes  $v = g \circ f(u, Y)$  from scratch. Similarly, when  $\mathcal{A}$  sends delete  $y$  query,  $\mathcal{S}$  removes  $y$  from set  $Y$  and computes  $v = g \circ f(u, Y)$ . Note that both operations do not require the auxiliary information. Finally,  $\mathcal{A}$  outputs an element  $y \notin Y$  (resp.  $y \in Y$ ) and a membership witness  $w$  (resp. nonmembership witness  $w$ ) for  $y$ .  $\mathcal{S}$  outputs  $w, y, Y$  and break the security property of the underlying UA.  $\square$

### A.2 Proof of Theorem 2

**Proof 2** Let  $\mathcal{A}$  be a polynomial-time adversary to our UA such that the maximum number of elements to be accumulated is  $k$ , we show how to construct a PPT simulator  $\mathcal{S}$  which solves the  $k$ -SDH problem in  $\mathbb{G}_1$  by invoking  $\mathcal{A}$ .

We assume the problem instance is from a restricted class of the  $q$ -SDH problem (specifically, the  $k$ -SDH problem instance is in a group of safe-prime order equipped with a bilinear map). Let  $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$  such that  $\mathbb{G}_1 = \langle g_0 \rangle$  and  $|\mathbb{G}_1| = p$  such that  $p = 2q + 1$  and both  $p, q$  are primes.  $\mathcal{S}$  is given  $\mathbb{G}_1, \mathbb{G}_2, \hat{e}, g_0, g_0^\alpha, \dots, g_0^{\alpha^k}$  and its goal is to output  $w^*, y^*$  such that  $w^{\alpha+y^*} = g_0$ . Such pair satisfy the relation  $\hat{e}(w^*, g_0^{y^*} g_0^\alpha) = \hat{e}(g_0, g_0)$ .

Let  $k$  be the maximum number of elements to be accumulated in the accumulator.  $\mathbb{G}_1, \mathbb{G}_2, \hat{e}, g_0, g_0^\alpha, \dots, g_0^{\alpha^k}$ , is given to the adversary as the definition of  $g \circ f$ . That is,  $f : (u, y) \mapsto u(\alpha + y)$  and  $g : y \mapsto g_0^y$  such that  $g \circ f(1, Y)$  is efficiently computable for any set  $Y \subset \mathbb{G}_q$  with  $|Y| \leq k$ .

If  $P_1$  is non-negligible,  $\mathcal{A}$  outputs a set  $Y \subset \mathbb{Z}_p$ , a value  $y' \notin Y$ , and a witness  $w$  such that  $\Omega(w, y', g \circ f(1, Y)) = 1$  with non-negligible probability. We have  $\hat{e}(w', g_0^{y'} g_0^\alpha) = \hat{e}(g \circ f(1, Y), g_0)$ . It is safe to assume  $\alpha \notin Y$ , otherwise  $\mathcal{S}$  solves the  $k$ -SDH problem immediately. Let  $v(\alpha) = \prod_{y \in Y} (\alpha + y)$ . Note that  $g \circ f(1, Y) = g_0^{v(\alpha)}$ . Since  $y' \notin Y$ , there exists a polynomial  $q(\alpha)$  of degree less than  $k$  such that  $v(\alpha) = q(\alpha)(y' + \alpha) + d$  for a constant  $d$ . Let  $w^* = [w' g_0^{-q(\alpha)}]^{1/d}$ .  $w^*$  is computable because  $\mathcal{S}$  can always express  $q(\alpha)$  as  $\sum_{i=1}^{i=k-1} u_i \alpha^i$ .  $\mathcal{S}$  sets  $y^* = y'$  and returns  $(w^*, y^*)$  as the solution to the  $k$ -SDH problem.

If  $P_2$  is non-negligible,  $\mathcal{A}$  outputs a set  $Y \subset \mathbb{Z}_p$ , a value  $y' \in Y$ , and a witness  $\bar{w} = (c', d') \in \mathbb{G}_1 \times \mathbb{Z}_p^*$  such that  $\bar{\Omega}(\bar{w}, y', g \circ f(1, Y)) = 1$  with non-negligible probability.

That is,  $\hat{e}(c', g_0^y g_0^\alpha) \hat{e}(g_0, g_0)^{d'} = \hat{e}(\mathbf{g} \circ \mathbf{f}(1, Y), g_0)$ . Similarly, it is safe to assume  $\alpha \notin Y$ , otherwise  $\mathcal{S}$  solves the  $k$ -SDH problem immediately. Let  $v(\alpha) = \prod_{y \in Y} (\alpha + y)$ . Since  $y' \in Y$ , there exists a polynomial  $q(\alpha)$  such that  $v(\alpha) = q(\alpha)(y' + \alpha)$ .  $\mathcal{S}$  compute  $w^* = [c' g_0^{-q(\alpha)}]^{-\frac{1}{d}}$ , sets  $y^* = y'$  and returns  $(w^*, y^*)$  as the solution to the  $k$ -SDH problem.  $\square$

### A.3 Proof of Theorem 3

**Proof 3** In order to prove the security of our ABACS, we must present a simulator  $\mathcal{S}$  satisfying Definition 5. We describe such a simulator and give a sketch of a proof that this simulator satisfies the definition.

The simulator  $\mathcal{S}$  represents the adversary-controlled parties to the system in the ideal world and represents the honest parties to the adversary in the real world. In the random oracle model,  $\mathcal{S}$  also get control over hash functions. Specifically,  $\mathcal{S}$  is described below.

**System Parameter and Hash Query** We assume  $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$  such that  $|\mathbb{G}_1| = |\mathbb{G}_2| = p$  and  $\mathbb{G}_q \subset \mathbb{Z}_p^*$  with  $|\mathbb{G}_q| = q$  are system-wide parameter. Let  $h$  be a generator of  $\mathbb{G}_q$  and  $H : \{0, 1\}^* \rightarrow \mathbb{G}_q$  be a hash function. Recalled that for each ACA  $O_j$ ,  $h_j$  is defined as  $x \mapsto h_j^x$  such that  $h_j = H(A_j)$  for some cryptographic hash function  $H$ , where  $A_j$  is the attribute managed by  $O_j$ . For each hash query, say  $H(A_j)$ ,  $\mathcal{S}$  randomly picks  $\rho_j \in_R \mathbb{Z}_q$  and returns  $h_j = H(A_j) = h^{\rho_j}$ . For each honest ACA in the ideal world,  $\mathcal{S}$  setup the public keys (DUA) for each of them in the real world honestly.

### Representing adversary-controlled users in the ideal world and honest players to adversary-controlled user in the real world

(Nym) When an adversary-controlled user initiates Nym by presenting pseudonym  $y_{i,j}$  to ACA  $O_j$ ,  $\mathcal{S}$  searches through the history to locate user  $i$ . Specifically,  $\mathcal{S}$  maintains a list of  $\mathcal{L}_A$  of adversary-controlled user. The list is indexed by value  $y_i$  defined as  $(y_{i,j})^{1/\rho_j}$ . If  $i$  is not found,  $\mathcal{S}$  adds  $y_i = (y_{i,j})^{1/\rho_j}$  to  $\mathcal{L}_A$  and contacts  $\mathcal{T}$  requesting a pseudonym with  $O_j$  on behalf of a new user  $U_i$ . Otherwise,  $\mathcal{S}$  contacts  $\mathcal{T}$  as an existing user  $U_i$ . In both cases,  $\mathcal{T}$  chooses a pseudonym  $\tilde{y}_{i,j}$  as the pseudonym. After that,  $\mathcal{S}$  appends  $y_{i,j}, \tilde{y}_{i,j}$  to the row indexed by  $y_i$  as the pseudonym between user  $U_i$  and  $y_{i,j}$  in the real world, and  $\tilde{y}_{i,j}$  is the corresponding pseudonym in the ideal world.

(CertifyAttribute) When an adversary-controlled user initiates CertifyAttribute by presenting pseudonym  $y_{i,j}$  to organization  $O_j$ ,  $\mathcal{S}$  searches through the history to locate user  $i$ . If  $y_{i,j}$  is not found, returns “fail”. Otherwise,  $\mathcal{S}$  contacts  $\mathcal{T}$  on behalf of  $U_i$  to  $O_j$ . If  $O_j$  in the ideal world reply with “accept”,  $\mathcal{S}$  returns “accept” to the adversary in the real world.  $\mathcal{S}$  updated the list  $\mathcal{L}_j$  in the real world as well.

(Authentication) When an adversary-controlled user is involved with Authentication with an honest verifier,  $\mathcal{S}$  first examines the policy  $ST$  to see if it involve any

possession of pseudonym. If yes, it locate user  $U_i$  using the list  $\mathcal{L}_A$  and contact  $\mathcal{T}$  for the corresponding Authentication functionality on behalf of  $U_i$ . If it does not involve any pseudonym,  $\mathcal{S}$  pick one  $U_i$  from the list  $\mathcal{L}_A$  that satisfy the policy if the authentication in the real world is successful. The simulation fails if the authentication in the real world is successful yet it fails in the ideal world or  $\mathcal{S}$  cannot found out any user in  $\mathcal{L}_A$  satisfy the policy.

### Representing adversary-controlled ACAs in the ideal world and honest players to ACAs in the real world

(Nym) When  $\mathcal{T}$  contacts the adversary-controlled ACA  $O_j$  in the ideal world with a pseudonym  $\tilde{y}_{i,j}$ ,  $\mathcal{S}$  computes  $y_{i,j} = h_j^x$  for some random  $x$ . Note that in our model,  $\mathcal{T}$  will not contact  $O_j$  if a pseudonym has previously setup between the underlying user and  $O_j$ .  $\mathcal{S}$  stores  $\tilde{y}_{i,j}$  and  $y_{i,j}$  as a pair in another list  $\mathcal{L}_H$ .

(CertifyAttribute) Using the list  $\mathcal{L}_H$ ,  $\mathcal{S}$  locates the corresponding user and represent it in the real world. If the adversary-controlled ACA  $O_j$  returns “accept”,  $\mathcal{S}$  notifies  $\mathcal{T}$  with “accept”.

### Representing adversary-controlled Verifiers in the ideal world and honest players to Verifiers in the real world

(Authentication) When  $\mathcal{T}$  contacts  $\mathcal{S}$  in the ideal world with a policy  $ST$ ,  $\mathcal{S}$  acts accordingly. In particular, if  $\mathcal{T}$  states that the underlying user satisfies the policy,  $\mathcal{S}$  uses the zero-knowledge simulator to simulate the Authentication protocol. This requires controlling the random oracle.  $\mathcal{S}$  then forwards the response of the adversary-controlled verifier in the real world back to  $\mathcal{T}$ . On the other hand, if  $\mathcal{T}$  states that the underlying user does not satisfy the policy,  $\mathcal{S}$  sends some incomplete authentication request to the adversary-controlled verifier and forwards its response back to  $\mathcal{T}$ .

**Successful Simulation** Due to the zero-knowledge nature of our protocols, the simulated Authentication protocols are perfect. Note that, however, the pseudonym formed by  $\mathcal{S}$  for honest users in the real world is not correct due to the fact that  $\mathcal{S}$  simply chooses  $x$  randomly each time. In particular, same underlying user might use different  $x$  to different ACAs. However, under the DDH assumption, adversary will not notice such difference. The simulation also fails if the authentication (of an adversary-controlled user) in the real world (to an honest verifier) is successful yet it fails in the ideal world. This only happens when the adversary is able to fake a proof during the Authentication protocol and happens with negligible probability provided that our DUA is secure. Thus, the simulator  $\mathcal{S}$  satisfies Definition 5 under the DDH assumption and the  $k$ -SDH assumption.  $\square$

## B Instantiation of PK<sub>4</sub>

In addition to  $\mathfrak{g}_0$  and  $\mathfrak{g}_1$ , let  $\mathfrak{g}_2, \mathfrak{g}_3, \mathfrak{g}_4$  be independent generators of  $\mathbb{G}_1$ . The prover computes auxiliary commitments  $\mathfrak{A}_1 = \mathfrak{g}_0^{\beta_1} \mathfrak{g}_1^{\beta_2} \in \mathbb{G}_1$ ,  $\mathfrak{A}_2 = c \mathfrak{g}_1^{\beta_1}$ ,  $\mathfrak{A}_3 = \mathfrak{g}_2^{\beta_3} \mathfrak{g}_3^{\beta_4}$ ,  $\mathfrak{A}_4 = \mathfrak{g}_4^{d\beta_3}$  for some randomly generated  $\beta_1, \beta_2, \beta_3, \beta_4 \in_R \mathbb{Z}_p^*$ . The prover sends  $\mathfrak{A}_1, \mathfrak{A}_2, \mathfrak{A}_3, \mathfrak{A}_4$  to the verifier and conducts the following proof.

$$\text{PK}_{4I} \left\{ (d, y, r, \beta_1, \beta_2, \beta_3, \beta_4, \delta_1, \delta_2, \delta_3, \delta_4) : \right.$$

$$\begin{aligned} \mathfrak{A}_1 &= \mathfrak{g}_0^{\beta_1} \mathfrak{g}_1^{\beta_2} \wedge 1 = \mathfrak{A}_1^{-y} \mathfrak{g}_0^{\delta_1} \mathfrak{g}_1^{\delta_2} \wedge \mathfrak{A}_3 = \mathfrak{g}_2^{\beta_3} \mathfrak{g}_3^{\beta_4} \wedge \\ 1 &= \mathfrak{A}_3^{-d} \mathfrak{g}_2^{\delta_3} \mathfrak{g}_3^{\delta_4} \wedge \mathfrak{A}_4 = \mathfrak{g}_4^{\delta_3} \wedge \mathfrak{C} = \mathfrak{g}_0^y \mathfrak{g}_1^r \wedge \\ &\quad \frac{\hat{e}(\mathfrak{A}_2, \mathfrak{g}_1)}{\hat{e}(v, \mathfrak{g}_0)} = \hat{e}(g_0, g_0)^d \hat{e}(\mathfrak{g}_1, g_1)^{\beta_1} \hat{e}(\mathfrak{g}_1, g_0)^{\delta_1} \hat{e}(\mathfrak{A}_2, g_0)^{-y} \end{aligned} \left. \right\}$$

The verifier also needs to check that  $\mathfrak{A}_4 \neq 1$ , which implies that  $d \neq 0$ .