Anada *J Surveill Secur Saf* 2024;5:160-83 **DOI:** 10.20517/jsss.2024.08

Journal of Surveillance, Security and Safety

Research Article

Open Access

Decentralized multi-authority anonymous credential system with bundled languages on identifiers in bilinear groups

Hiroaki Anada[1,2]

[1]Department of Software and Information Technology, Aomori University, Aomori 030-0943, Japan.
[2]Department of Mathematical Informatics, Meiji Gakuin University, Yokohama 244-8539, Japan.

Correspondence to: Prof. Hiroaki Anada, Department of Mathematical Informatics, Meiji Gakuin University, 1518, Kamikurata-cho, Totsuka-ku, Yokohama 244-8539, Japan. E-mail: hiroaki.anada@mi.meijigakuin.ac.jp

How to cite this article: Anada H. Decentralized multi-authority anonymous credential system with bundled languages on identifiers in bilinear groups. *J Surveill Secur Saf* 2024;5:160-83. http://dx.doi.org/10.20517/jsss.2024.08

Received: 28 Mar 2024 **First Decision:** 19 Jul 2024 **Revised:** 2 Sep 2024 **Accepted:** 2 Sep 2024 **Published:** 20 Sep 2024

Academic Editors: Qiong Huang, Josef Pieprzyk **Copy Editor:** Dong-Li Li **Production Editor:** Dong-Li Li

Abstract

We propose a multi-show decentralized multi-authority attribute-based anonymous credential system (dACS). Referring to previous work, we give a new syntax and three security notions: unforgeability, anonymity and unlinkability. Especially, *corruption* of authorities is considered to reflect a real scenario. Then we give a generic construction of dACS. In our dACS, an attribute authority who issues a private secret key to an entity only has to sign the entity's identifier. Then, according to the principle of "commit-to-identifier", the entity generates a proof of knowing credentials. There are two building blocks: the structure-preserving signature scheme and the Groth-Sahai non-interactive proof system, both of which are in asymmetric bilinear groups. The principle is realized with a bundled language that is simultaneous pairing-product equations on the identifier. There, the bundled language works for preventing *collusion attacks*. Finally, we instantiate our generic dACS under the Symmetric External Diffie-Hellman (SXDH) assumption, compare the instantiated scheme with previous work, and evaluate the performance.

Keywords: Anonymous credential system, decentralized multi-authority, Groth-Sahai proofs, structure-preserving signatures

INTRODUCTION

A global identifier is a string of a digital identity that is linked to an entity in our cyberspace. An e-mail address issued by a reliable organization and a universally unique identifier (UUID) stipulated by ISO/IEC 11578:1996[1] can be a global identifier. Global identifiers are registered by authorities and used by entities to execute some rights in the space. An anonymous credential system (ACS) that was first proposed by Chaum is a system in which an entity with an identifier is given a credential of a right issued by an authority[2]. Then the entity can prove possession of the credential to a verifier, which is typically a service provider, without leaking its identity information. Thus, a primary aim of ACS is privacy protection in transactions in which a right of an entity is checked.

Towards real applications, ACS has been studied for efficiency in the mathematical structures of Rivest-Shamir-Adleman (RSA), discrete logarithm, bilinear groups, lattices, *etc.*[3–6]. As for functions of ACS, whether anonymous credentials are single-show or multi-show[3] is critical. A single-show ACS was introduced firstly by Brands[7], in which a credential can be proven only once; if it is proven more than once, then those proofs are possibly linked to avoid double spending. On the other hand, a multi-show ACS was introduced firstly by Camenisch-and-Lysyanskaya[3], in which a credential can be proven more than once keeping unlinkability. Another function of importance is to treat attribute credentials. Tan-and-Groß[8] introduced an attribute-based ACS (abACS), in which an entity can prove possession of a number of credentials simultaneously. For instance, it can prove possession of its attributes such as age = 30 AND gender = female AND nationality = USA. Further, Chan and Yuen developed an attributed-based ACS which supports both single-show and multi-show selectively[9]. In the design of such abACS, a primary target is efficiency from the viewpoints of computational amount and data length of a proof that an entity is in possession of claimed attribute credentials. Since a naive construction with linear complexity is easy because a simultaneous showing of their proofs suffices the need, asymptotic behavior smaller than linear complexity was pursued[8,9]. In contrast, a decentralized multi-authority ACS (dACS) that was introduced by Garman *et al.* is a different direction of study[10]. In a dACS, there are a number of authorities of issuing attribute credentials, and there is no central authority among them. Each authority is responsible for each attribute, and once a global identifier is linked to an entity, the authority is able to issue its attribute credential to the identifier.

A challenging task in the design of dACS is to attain collusion resistance. That is, in the case of dACS, the verifier should resist collusion attacks by adversaries who bring together their attribute credentials issued to different global identifiers. Note that the collusion resistance has been studied in attribute-based cryptographic primitives such as attribute-based encryption[11] and signatures[12], but in the case of dACS, it has not been studied yet. Another challenging task is to design dACS so that it is capable of treating any given formula for fine-grained access control, such as a monotone formula over attributes. Actually, the notion of "attributed-based" was initially introduced in the case of encryption and decryption by Sahai and Waters[13], and was developed by the subsequent work by, for example, Goyal *et al.*[14], Chase-and-Chow[15], *etc.* The anonymity, collusion resistance and fine-grained access control are three properties that need appropriate (and subtle) design techniques.

Our contribution

In this paper, we propose a multi-show dACS, which is able to treat any given all-AND formula. We first give syntax of our dACS. Then we give three security notions. One is existential unforgeability (EUF) against collusion attacks. There, we introduce corruption of authorities reflecting a real scenario that an adversary can corrupt some of the authorities and get their master secret keys. The second and third are anonymity and unlinkability of proofs. In our definitions, anonymity means that any probabilistic polynomial-time (PPT) adversary including the issuer can get only a negligible amount of information on identifiers from given proofs. On the other hand, unlinkability means that any PPT adversary cannot distinguish two cases; the first case is that two given proofs are generated by a single entity and the second case is that the two proofs are generated by

two entities with different identifiers. Thus, the unlinkability of proofs is a stronger notion than the anonymity in our definitions, and we will prove the implication.

We then give a generic construction of dACS. In our dACS, a feature is that an attribute authority who issues a private secret key to an entity only has to sign the entity's identifier. At that time, the authority uses a set of common public parameters in a standard like NIST FIPS 186-4[16]. Then, according to the principle of "commit-to-identifier", the entity generates a proof of knowing credentials. There are two building blocks in the construction. One is the structure-preserving signature scheme[17] and the other is the Groth-Sahai non-interactive proof system[18,19], where both blocks are based on Type 3 asymmetric bilinear groups[20]. In the construction, the principle is realized with a bundled language that is simultaneous pairing-product equations on the entity's identifier and a structure-preserving signature on the identifier. Thus, the bundled language works for preventing collusion attacks. The bundled language is a special case of simultaneous equation systems. It would be natural to consider a generalization into the case of more than one common variable. The study of this direction is of independent interest.

In the succeeding section, we instantiate our generic dACS under the Symmetric External Diffie-Hellman (SXDH) assumption[17–19] on the Type 3 pairings. Then we compare features of our instantiated $\mathrm{dACS}_{\mathrm{sxdh}}$ with previous work and evaluate efficiency. As a result, it turns out that "decentralized multi-authority, security in the standard model and unforgeability under the (simple) SXDH assumption" is a positive aspect, as well as security considering partial corruption of authorities. Efficiency evaluation of our $\mathrm{dACS}_{\mathrm{sxdh}}$ shows that, when the number of attribute credentials involved in a proof is two, the data length of a proof is 68k bytes, and generation and verification times are 2.8 and 1.8 s, respectively. Since the proof size is linear in the number of attribute credentials involved in a proof, a construction with smaller asymptotic behavior should be our future work.

Related recent work

From the viewpoint of issuing authorities, the work by Garman *et al.* is the first ACS with decentralized multi-authority in the attribute-based setting (dACS, for short), which is capable of treating all-AND formulas[10]. Our dACS, in addition, is proven secure even when some of the authorities are corrupted (i.e., the master secret keys of them are leaked to adversaries).

Collusion attacks are also considered in the work by Garman *et al.*, but the security claim is in the random oracle model[10]. As for collusion resistance in the standard model, Camenisch *et al.* proposed a dACS that has the property[21]. Moreover, the ACS[21] has the security of the universal composability[22,23]. Our $\mathrm{dACS}_{\mathrm{sxdh}}$ instantiated under the SXDH assumption is also universally composable due to the universal composability of the Groth-Sahai proof system[18,19]. We note that, though the dACS[21] and our $\mathrm{dACS}_{\mathrm{sxdh}}$ attain similar properties, the ACS by Camenisch *et al.* needs more assumptions than our $\mathrm{dACS}_{\mathrm{sxdh}}$ in the security proof of unforgeability[21].

As for fine-grained access control, the abACS by Sadiah *et al.* is capable of treating monotone formulas, though the proof size is exponential to the number of attributes appearing in the formula[24]. The abACS by Okishima-and-Nakanishi[25] is capable of treating Conjunctive Normal Form (CNF) formulas. The abACS by Fuchsbauer *et al.* is capable of treating all-and formulas with an advantage of prover anonymity at issuing phase[26]. We note that these three abACSs have not been studied in a security experiment of collusion resistance. Though the two abACSs by Tan and Groß[8] and Chan and Yuen[9] mentioned above are capable of treating monotone formulas and have collusion resistance, they are not in the setting of decentralized multi-authority. Concerning the attributes issued under each authority, our dACS can treat any access policy of an all-AND formula that covers plural attributes for each authority (and the number of authorities is more than one). From the viewpoint, designing a decentralized multi-authority abACS with more fine-grained access policies, for instance, any monotone formulas, is a challenging problem.

Replay attack is one of the most typical threats to authentication systems. In a replay attack, an attacker intercepts valid data that are transmitted from a prover to a verifier. Then it maliciously re-sends them, after some time intervals, to make the verifier accept the prover. Thus, the aim of a replay attack is to cause mis-authentication that leads to potential stronger security breaches. It is also possibly applicable to ACSs, especially because of anonymity. One of the typical countermeasures against replay attack is introducing interactive proofs. That is, the verifier generates a random challenge message at every session of authentication to reject a re-sent response message. As for non-interactive proofs, in recent privacy-preserving systems[27,28] in which proofs are non-interactive, permissioned blockchains are employed, which is in contrast with the permissionless blockchain employed in Bitcoin[29]. A permissioned blockchain fits our dACS because transactions including anonymous credentials are permitted by the authorities. In the systems, the replay attack can be detected by the authorities.

Finally, we note here recent studies that developed more functions than our dACS. Au *et al.* proposed a dynamic k-times anonymous authentication (k-TAA) scheme, which allows members of a group to be authenticated anonymously by application providers for a bounded number of times, where application providers can independently and dynamically grant or revoke access rights to members in their own group[30]. Concerning a revocation mechanism that is needed for real usage, Ma and Chow[31] proposed updatable anonymous credentials with revocation and reputation management. Extending these concepts to multi-authority systems (as discussed in its appendix) would provide a broader context for our dACS. As for threshold signatures, Doerner *et al.* proposed an ACS with threshold issuance by constructing a secure multiparty signing protocol for the BBS+ signature scheme[32]. Wong *et al.* proposed a secure multiparty computation of threshold signatures, which presented an efficient threshold Elliptic Curve Digital Signature Algorithm (ECDSA) protocol[33].

Our work in this paper is a significantly extended version of the proceeding paper presented at SecITC 2020[34]. Especially the sections of "Introduction", "Instantiation" and "Feature Comparison and Efficiency Evaluation" are totally expanded.

Organization of the paper

In Section "PRELIMINARIES", we fix notations and summarize the needed notions for later sections. In Section "BUNDLED LANGUAGE", we explain our ideas, which is for the Groth-Sahai proofs. In Section "DECENTRALIZED MULTI-AUTHORITY ANONYMOUS CREDENTIAL SYSTEM", we propose the syntax and security definitions of our dACS. In Section "GENERIC CONSTRUCTION", we give a construction of our dACS employing the Groth-Sahai proof system and a structure-preserving signature scheme. In Section "INSTANTIATION", we concretely describe our dACS under the SXDH assumption. In Section "FEATURE COMPARISON AND EFFICIENCY EVALUATION", we compare the features of our instantiated $\mathrm{dACS}_{\mathrm{sxdh}}$ with those of the previous abACSs. Then we evaluate efficiency of our $\mathrm{dACS}_{\mathrm{sxdh}}$ by partial implementation and estimation. In Section "CONCLUSION", we summarize our work and state future directions.

PRELIMINARIES

$\mathbb{N}$ denotes the set of natural numbers. $[n]$ represents the subset $\{1, \ldots, n\} \subset \mathbb{N}$. $\mathbb{Z}_p$ indicates the residue class ring of integers modulo a prime number p. λ stands for the security parameter, where $\lambda \in \mathbb{N}$. A function $P(\lambda)$ is said to be negligible in λ if for any given positive polynomial $\mathrm{poly}(\lambda)$ $P(\lambda) < 1/\mathrm{poly}(\lambda)$ for sufficiently large λ. Two functions $P(\lambda)$ and $Q(\lambda)$ are said to be computationally indistinguishable in λ if $|P(\lambda) - Q(\lambda)|$ is negligible in λ, which we denote $P(\lambda) \approx_{\mathrm{c}} Q(\lambda)$. $a \in_R S$ denotes a uniform random sampling of an element a from a set S. $a =_? b$ points to a boolean decision, which returns 1 if $a = b$ and 0 otherwise. $z \leftarrow A(a; r)$ denotes that z is returned by a probabilistic algorithm A with an input a and a randomness r on a random

tape. *St* corresponds to the inner state of an algorithm. $(c_i)_i$ abbreviates a vector $c = (c_i)_{i\in I}$. Similarly, $(c^a)^a$ abbreviates a vector $c = (c^a)^{a\in A}$, and $(c_i^a)_i^a$ abbreviates a vector $c = (c_i^a)_{i\in I}^{a\in A}$.

Bilinear groups

Let $\mathcal{BG}$ be a generation algorithm of bilinear groups[20]: $\mathcal{BG}(1^\lambda) \to (p, \hat{\mathbb{G}}, \check{\mathbb{G}}, \mathbb{T}, e, \hat{G}, \check{G})$. Here p is a prime number of bit-length λ, $\hat{\mathbb{G}}, \check{\mathbb{G}}$ and $\mathbb{T}$ are cyclic groups of order p, and $\hat{G}$ and $\check{G}$ are generators of $\hat{\mathbb{G}}$ and $\check{\mathbb{G}}$, respectively. We denote operations in $\hat{\mathbb{G}}$, $\check{\mathbb{G}}$ and $\mathbb{T}$ multiplicatively. e is the bilinear map $\hat{\mathbb{G}} \times \check{\mathbb{G}} \to \mathbb{T}$. e should have the following two properties: Non-degeneracy: $e(\hat{G}, \check{G}) \neq 1_{\mathbb{T}}$, and Bilinearity: $\forall a \in \mathbb{Z}_p, \forall b \in \mathbb{Z}_p, \forall \hat{X} \in \hat{\mathbb{G}}, \forall \check{Y} \in \check{\mathbb{G}}, e(\hat{X}^a, \check{Y}^b) = e(\hat{X}, \check{Y})^{ab}$. Hereafter we denote an element in $\hat{\mathbb{G}}$ and $\check{\mathbb{G}}$ with hat " ˆ " and check " ˇ ", respectively. Then, according to the previous work by Escala and Groth[19], we introduce the following linear algebra-friendly additive notations.

$$\forall \hat{x}_1, \forall \hat{x}_2 \in \hat{\mathbb{G}} \ \ \hat{x}_1 + \hat{x}_2 \stackrel{\text{def}}{=} \hat{x}_1\hat{x}_2, \ \ \forall \check{y}_1, \forall \check{y}_2 \in \check{\mathbb{G}} \ \ \check{y}_1 + \check{y}_2 \stackrel{\text{def}}{=} \check{y}_1\check{y}_2, \tag{1}$$

$$\forall \hat{x} \in \hat{\mathbb{G}} \forall a \in \mathbb{Z}_p \ \ \hat{x}a \stackrel{\text{def}}{=} \hat{x}^a, \ \ \forall \check{y} \in \check{\mathbb{G}}, \forall b \in \mathbb{Z}_p \ \ b\check{y} \stackrel{\text{def}}{=} \check{y}^b, \tag{2}$$

$$\forall \hat{x} \in \hat{\mathbb{G}}, \forall \check{y} \in \check{\mathbb{G}} \ \ \hat{x} \cdot \check{y} \stackrel{\text{def}}{=} e(\hat{x}, \check{y}), \tag{3}$$

$$\forall z_1, \forall z_2 \in \mathbb{T} \ \ z_1 + z_2 \stackrel{\text{def}}{=} z_1 z_2. \tag{4}$$

Then, for further simplicity, we introduce the following notation.

$$\forall \hat{x} \in \hat{\mathbb{G}}, \forall a \in \mathbb{Z}_p, \forall \check{y} \in \check{\mathbb{G}} \ \ \hat{x}a\check{y} \stackrel{\text{def}}{=} \hat{x}a \cdot \check{y} = \hat{x} \cdot a\check{y}. \tag{5}$$

Then it is easy to see that the following equality holds.

$$\begin{aligned} &\forall \hat{x}_1, \forall \hat{x}_2 \in \hat{\mathbb{G}}, \forall a, \forall b, \forall c, \forall d \in \mathbb{Z}_p, \forall \check{y}_1, \forall \check{y}_2 \in \check{\mathbb{G}} \\ &(\hat{x}_1, \hat{x}_2) \begin{pmatrix} ac & ad \\ bc & bd \end{pmatrix} (\check{y}_1, \check{y}_2)^\top = e(\hat{x}_1^a \hat{x}_2^b, \check{y}_1^c \check{y}_2^d) \ \in \mathbb{T}. \end{aligned} \tag{6}$$

Finally, we extend the notation [Equation (3)] to a vector and a matrix form in the following way.

$$\begin{aligned} &\forall \hat{x}_1, \forall \hat{x}_2 \in \hat{\mathbb{G}}, \forall \check{y}_1, \forall \check{y}_2 \in \check{\mathbb{G}} \\ &\begin{pmatrix} \hat{x}_1 \\ \hat{x}_2 \end{pmatrix} \cdot (\check{y}_1, \check{y}_2) = \begin{pmatrix} e(\hat{x}_1, \check{y}_1) & e(\hat{x}_1, \check{y}_2) \\ e(\hat{x}_2, \check{y}_1) & e(\hat{x}_2, \check{y}_2) \end{pmatrix} \in \mathbb{T}^{2\times 2}, \end{aligned} \tag{7}$$

and

$$\begin{aligned} &\forall \hat{x}_1, \forall \hat{x}_2, \forall \hat{x}_3, \forall \hat{x}_4 \in \hat{\mathbb{G}}, \forall \check{y}_1, \forall \check{y}_2, \check{y}_3, \forall \check{y}_4 \in \check{\mathbb{G}} \\ &\begin{pmatrix} \hat{x}_{1,1} & \hat{x}_{1,2} \\ \hat{x}_{2,1} & \hat{x}_{2,2} \end{pmatrix} \cdot \begin{pmatrix} \check{y}_{1,1} & \check{y}_{1,2} \\ \check{y}_{2,1} & \check{y}_{2,2} \end{pmatrix} = \begin{pmatrix} e(\hat{x}_{1,1}, \check{y}_{1,1})e(\hat{x}_{1,2}, \check{y}_{2,1}) & e(\hat{x}_{1,1}, \check{y}_{1,2})e(\hat{x}_{1,2}, \check{y}_{2,2}) \\ e(\hat{x}_{2,1}, \check{y}_{1,1})e(\hat{x}_{2,2}, \check{y}_{2,1}) & e(\hat{x}_{2,1}, \check{y}_{1,2})e(\hat{x}_{2,2}, \check{y}_{2,2}) \end{pmatrix} \in \mathbb{T}^{2\times 2}. \end{aligned} \tag{8}$$

Structure-preserving signature scheme

The structure-preserving signature scheme[17,35] Sig consists of four PPT algorithms: $\mathsf{Sig} = (\mathsf{Sig.Setup}, \mathsf{Sig.KG}, \mathsf{Sig.Sign}, \mathsf{Sig.Vrf})$.

$\mathsf{Sig.Setup}(1^\lambda) \to pp$. On input the security parameter 1^λ, this PPT algorithm executes the generation algorithm of bilinear groups, and it sets the output as a set of public parameters: $\mathcal{BG}(1^\lambda) \to (p, \hat{\mathbb{G}}, \check{\mathbb{G}}, \mathbb{T}, e, \hat{G}, \check{G}) =: pp$. It returns pp.

$\mathsf{Sig.KG}() \to (\mathrm{PK}, \mathrm{SK})$. Based on the set of public parameters pp, this PPT algorithm generates a signing key SK and the corresponding public key PK. It returns (PK, SK).

$\mathsf{Sig.Sign}(\mathrm{PK}, \mathrm{SK}, m) \to \sigma$. On input the public key PK, the secret key SK and a message $m \in \hat{\mathbb{G}}$ or $\check{\mathbb{G}}$, this PPT algorithm generates a signature σ. In the case of the structure-preserving signatures, σ consists of elements $(V_i)_i$ where V_i is in either $\hat{\mathbb{G}}$ or $\check{\mathbb{G}}$. It returns $\sigma := (V_i)_i$.

$\mathsf{Sig.Vrf}(\mathrm{PK}, m, \sigma) \to d$. On input the public key PK, a message $m \in \hat{\mathbb{G}}$ or $\check{\mathbb{G}}$ and a signature $\sigma = (V_i)_i$, this deterministic algorithm returns a boolean decision d.

The correctness should hold for the scheme Sig: For any security parameter 1^λ, any set of public parameters $pp \leftarrow \mathsf{Sig.Setup}(1^\lambda)$ and any message m, $\Pr[d = 1 \mid (\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Sig.KG}(), \sigma \leftarrow \mathsf{Sig.Sign}(\mathrm{PK}, \mathrm{SK}, m), d \leftarrow \mathsf{Sig.Vrf}(\mathrm{PK}, m, \sigma)] = 1$.

Adaptive chosen-message attack of an existential forgery on the scheme Sig by a forger algorithm $\mathbf{F}$ is defined by the following algorithm of an experiment.

$$
\begin{aligned}
&\mathsf{Exp}^{\text{euf-cma}}_{\mathsf{Sig},\mathbf{F}}(1^\lambda):\\
&\quad pp \leftarrow \mathsf{Sig.Setup}(1^\lambda), (\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Sig.KG}()\\
&\quad (m^*, \sigma^*) \leftarrow \mathbf{F}^{\mathbf{SignO}(\mathrm{PK},\mathrm{SK},\cdot)}(pp, \mathrm{PK})\\
&\quad \text{If } m^* \notin \{m_j\}_{1 \le j \le q_s} \text{ and } \mathsf{Sig.Vrf}(\mathrm{PK}, m^*, \sigma^*) = 1,\\
&\qquad \text{then Return Win else Return Lose}
\end{aligned}
$$

In the above experiment, $\mathbf{F}$ issues a query m_j to its signing oracle $\mathbf{SignO}(\mathrm{PK}, \mathrm{SK}, \cdot)$. Then $\mathbf{F}$ receives a valid signature σ_j as a reply. The queries are at most q_s times ($1 \le j \le q_s$), and are bounded by a polynomial in λ. After receiving the replies, $\mathbf{F}$ returns a message-signature pair (m^*, σ^*). A restriction is imposed on $\mathbf{F}$ that the message m^* of the forgery should not be contained in the set of queries $\{m_j\}_{1 \le j \le q_s}$. The advantage of $\mathbf{F}$ over Sig is defined as $\mathbf{Adv}^{\text{euf-cma}}_{\mathsf{Sig},\mathbf{F}}(\lambda) := \Pr[\mathsf{Exp}^{\text{euf-cma}}_{\mathsf{Sig},\mathbf{F}}(1^\lambda) \text{ returns Win}]$.

Definition 1 (EUF-CMA[36]) *The scheme* Sig *is said to be* existentially unforgeable against adaptive chosen-message attacks (EUF-CMA) *if, for any PPT algorithm* $\mathbf{F}$*, the advantage* $\mathbf{Adv}^{\textit{euf-cma}}_{\mathsf{Sig},\mathbf{F}}(\lambda)$ *is negligible in* λ.

Non-interactive commit-and-prove scheme for structure-preserving signatures

According to the "fine-tuning Groth-Sahai proofs" system[19], we survey here the non-interactive commit-and-prove scheme on pairing-product equations, though we treat them in their additive forms. A commit-and-prove scheme CmtPrv consists of six PPT algorithms: $\mathsf{CmtPrv} = [\mathsf{CmtPrv.Setup}, \mathsf{Cmt} = (\mathsf{Cmt.KG}, \mathsf{Cmt.Com}, \mathsf{Cmt.Vrf}), \Pi = (\mathsf{Prv.P}, \mathsf{Prv.V})]$. Intuitively, there are three parts in CmtPrv and they function as follows. The first part $\mathsf{CmtPrv.Setup}$ generates a set of public parameters pp. The second, commit-part, provides a set of tools that realize a "cryptographic envelope" from a sender to a receiver, which is along the technique of "dual-mode" commitment[18]. The third, prove-part, is for a prover to generate a proof to a verifier, which is to prove that the prover knows the data committed by the commit-algorithm of the commit-part, in a witness-indistinguishable way.

Language

We first describe the language for which our scheme will work. The language is dependent on the type of verification equations of the Groth-Sahai proofs (group-dependent languages[18]). For this purpose, we first fix the set of public parameters.

• $\mathsf{CmtPrv.Setup}(1^\lambda) \to pp$. On input the security parameter 1^λ, this PPT algorithm executes a generation algorithm of bilinear groups $\mathcal{BG}$, and it sets the output as the public parameters pp: $\mathcal{BG}(1^\lambda) \to (p, \hat{\mathbb{G}}, \check{\mathbb{G}}, \mathbb{T}, e, \hat{G}, \check{G}) =: pp$. It returns pp.

Let $n \in \mathbb{N}$ be a constant. Suppose that we are given an equation system with n equations and with variables $(\hat{X}_i)_i$ and $(\check{Y}_j)_j$:

$$\begin{cases} \sum_i \hat{X}_i \cdot \check{B}_{1i} + \sum_j \hat{A}_{1j} \cdot \check{Y}_j + \sum_i \sum_j \hat{X}_i \gamma_{1ij} \check{Y}_j = t_{\mathbb{T}1}, \\ \vdots \\ \sum_i \hat{X}_i \cdot \check{B}_{ni} + \sum_j \hat{A}_{nj} \cdot \check{Y}_j + \sum_i \sum_j \hat{X}_i \gamma_{nij} \check{Y}_j = t_{\mathbb{T}n}. \end{cases} \tag{9}$$

Let L denote the set of coefficients of the equation system [Equation (9)] and $W(x)$ denote the set of solutions for $x \in L$:

$$L := \{x \in (\prod_i \hat{\mathbb{G}} \times \prod_j \check{\mathbb{G}} \times \prod_i \prod_j \mathbb{Z}_p)^n \mid x = ((\check{B}_{ki})_i, (\hat{A}_{kj})_j, (\gamma_{kij})_{i,j})_{k=1}^n\}, \tag{10}$$

$$W(x) := \{w \in \prod_i \hat{\mathbb{G}} \times \prod_j \check{\mathbb{G}} \mid w = ((\hat{W}_i)_i, (\check{W}_j)_j) \text{ satisfies (9) for } x\}, \tag{11}$$

$$R := \{(x, w) \in (\prod_i \hat{\mathbb{G}} \times \prod_j \check{\mathbb{G}} \times \prod_i \prod_j \mathbb{Z}_p)^n \times \prod_i \hat{\mathbb{G}} \times \prod_j \check{\mathbb{G}} \mid (x, w) = (((\check{B}_{ki})_i, (\hat{A}_{kj})_j, (\gamma_{kij})_{i,j})_{k=1}^n, ((\hat{W}_i)_i, (\check{W}_j)_j)) \text{ satisfies (9)}\}. \tag{12}$$

For a fixed parameter set pp, we call L, $W(x)$ and R the group-dependent language with pp, the witness space of x with pp and the relation with pp, respectively.

Commit-part

The commit-part[18,19] $\mathsf{Cmt} = (\mathsf{Cmt.KG}, \mathsf{Cmt.Com}, \mathsf{Cmt.Vrf})$ is described as follows.

• $\mathsf{Cmt.KG}(\mathtt{mode}) \to key$. On input a string `mode`, this PPT algorithm generates a *key*. If `mode = nor`, then $key = ck$ which is a commitment key. If `mode = ext`, then $key = (ck, xk)$ which is a pair of ck and an extraction key xk. If `mode = sim`, then $key = (ck, tk)$ which is a pair of ck and a trapdoor key tk. It returns *key*.

We put $pp := (pp, ck)$. Note here that the commitment key ck, which is a common reference string (CRS) in the term of non-interactive proof systems[18,19], is treated as one of the public parameters.

• $\mathsf{Cmt.Com}(w; r) \to (c, r)$. On input a message w (which will be a witness in the proof-part), this PPT algorithm generates a commitment c with a randomness r. r will also be a verification key. It returns (c, r). When w is a vector $w = (w_i)_i$, c and r are also vectors of the same number of components: $c = (c_i)_i$ and $r = (r_i)_i$. Note that computation is executed in the *componentwise way*; $\mathsf{Cmt.Com}(w_i; r_i) \to (c_i, r_i)$.

• $\mathsf{Cmt.Vrf}(c, w, r) \to d$. On input a commitment c, a message w and a verification key r, this deterministic algorithm generates a boolean decision d. It returns d.

The commit-part Cmt of the Groth-Sahai proof system has the four properties[19]: *(1) perfect correctness, (2) dual mode, (3) perfectly binding and (4) perfectly hiding*. We remark here that the properties (3) and (4) cannot stand simultaneously in principle (see "Commitment Scheme"[37], *etc.*). The trick is that, when the `mode` is `ext`, the *key* is (ck, xk) and the property (3) "perfectly binding" holds. When the `mode` is `sim`, the *key* is (ck, tk) and the property (4) "perfectly hiding" holds. Importantly for security proofs, the two modes of commitment keys (cks) are assumed to be computationally indistinguishable under an appropriate assumption. The detailed definitions are given in Definition 3.

Prove-part

The prove-part[18,19] $\Pi = (\mathsf{Prv.P}, \mathsf{Prv.V})$ is described as follows.

• $\mathsf{Prv.P}(x, c, w, r) \to \pi$. On input a statement x, a commitment c, a witness w and a randomness r which was used to generate a commitment c, this PPT algorithm executes the proof-generation algorithm of the Groth-Sahai proof system to obtain a proof π (see ref[19] for the details and ref[17,38] for instantiations). It returns π.

• $\mathsf{Prv.V}(x, c, \pi) \to d$. On input a statement x, a commitment c and a proof π, this deterministic algorithm executes the verification algorithm of the Groth-Sahai proof system to obtain a boolean decision d (see ref[19] for the details). It returns d.

The proof-part $(\mathsf{CmtPrv.Setup}, \Pi)$ of the Groth-Sahai proof system has the four properties[19]: *(1) perfect correctness, (2) perfect soundness, (3) perfect F-knowledge and (4) composable witness-indistinguishability [especially (4) means perfect witness-indistinguishability]*. The detailed definitions are given later.

Four properties of commit-part

Definition 2 (Correctness[18,19]) *A commitment scheme Cmt is said to be correct if it satisfies the following condition: For any security parameter 1^λ, any set of public parameters $pp \leftarrow \mathsf{CmtPrv.Setup}(1^\lambda)$, any commitment key $ck \leftarrow \mathsf{Cmt.KG}(\mathtt{mode})$ where $\mathtt{mode} = \mathtt{nor}$ or $\mathtt{ext}$ or $\mathtt{sim}$, and any message w,*

$$\Pr[d = 1 \mid (c, r) \leftarrow \mathsf{Cmt.Com}(w), d \leftarrow \mathsf{Cmt.Vrf}(c, w, r)] = 1.$$

Definition 3 (Dual Mode[18]) *A commitment scheme Cmt is said to be dual mode if it satisfies the following condition: For any security parameter 1^λ, any set of public parameters $pp \leftarrow \mathsf{CmtPrv.Setup}(1^\lambda)$ and any PPT algorithm $\mathbf{A}$,*

$$\Pr[\mathbf{A}(pp, ck) = 1 \mid ck \leftarrow \mathsf{Cmt.KG}(\mathtt{nor})] = \Pr[\mathbf{A}(pp, ck) = 1 \mid (ck, xk) \leftarrow \mathsf{Cmt.KG}(\mathtt{ext})], \quad (13)$$

$$\Pr[\mathbf{A}(pp, ck) = 1 \mid ck \leftarrow \mathsf{Cmt.KG}(\mathtt{nor})] \approx_c \Pr[\mathbf{A}(pp, ck) = 1 \mid (ck, tk) \leftarrow \mathsf{Cmt.KG}(\mathtt{sim})]. \quad (14)$$

From Equation (13), the computational indistinguishability [(Equation (14))] is equivalent to the following: For any security parameter 1^λ, for any set of public parameters $pp \leftarrow \mathsf{CmtPrv.Setup}(1^\lambda)$ and any PPT algorithm $\mathbf{A}$, the advantage $\mathbf{Adv}^{\text{ind-dual}}_{\mathsf{Cmt},\mathbf{A}}(\lambda)$ of $\mathbf{A}$ over Cmt defined by the difference below is negligible in λ:

$$\begin{aligned}\mathbf{Adv}^{\text{ind-dual}}_{\mathsf{Cmt},\mathbf{A}}(\lambda) \stackrel{\text{def}}{=} |&\Pr[\mathbf{A}(pp, ck) = 1 \mid (ck, xk) \leftarrow \mathsf{Cmt.KG}(\mathtt{ext})] \\ &- \Pr[\mathbf{A}(pp, ck) = 1 \mid (ck, tk) \leftarrow \mathsf{Cmt.KG}(\mathtt{sim})]|. \end{aligned} \quad (15)$$

The indistinguishability [Equation (15)] holds, for example, for an instance of the Groth-Sahai proof system under the SXDH assumption[18,19].

Definition 4 (Perfectly Binding[18]) *A commitment scheme Cmt is said to be perfectly binding if it satisfies the following condition for some unbounded algorithm $\mathsf{Cmt.Open}$: For any security parameter 1^λ, any set of public parameters $pp \leftarrow \mathsf{CmtPrv.Setup}(1^\lambda)$, any commitment key $ck \leftarrow \mathsf{Cmt.KG}(\mathtt{nor})$ and any message w,*

$$\Pr[w = w' \mid (c, r) \leftarrow \mathsf{Cmt.Com}(w; r), w' \leftarrow \mathsf{Cmt.Open}(c)] = 1.$$

Definition 5 (Perfectly Hiding[18]) *A commitment scheme Cmt is said to be perfectly hiding if it satisfies the following condition: For any security parameter 1^λ, any set of public parameters $pp \leftarrow \mathsf{CmtPrv.Setup}(1^\lambda)$, any commitment key ck s.t. $(ck, tk) \leftarrow \mathsf{Cmt.KG}(\mathtt{sim})$ and any PPT algorithm $\mathbf{A}$,*

$$\begin{aligned}&\Pr[\mathbf{A}(St, c) = 1 \mid (w, w', St) \leftarrow \mathbf{A}(pp, ck, tk), (c, r) \leftarrow \mathsf{Cmt.Com}(w)] \\ &= \Pr[\mathbf{A}(St, c') = 1 \mid (w, w', St) \leftarrow \mathbf{A}(pp, ck, tk), (c', r') \leftarrow \mathsf{Cmt.Com}(w')]. \end{aligned} \quad (16)$$

Four properties of prove-part

Definition 6 (Perfect Correctness[18]) *A commit-and-prove scheme* CmtPrv *is said to be perfectly correct if it satisfies the following condition: For any security parameter* 1^λ*, any set of public parameters* $pp \leftarrow \mathsf{CmtPrv.Setup}(1^\lambda)$*, any commitment key* $ck \leftarrow \mathsf{Cmt.KG}(\mathtt{mode})$ *where* $\mathtt{mode} = \mathtt{nor}$ *or* $\mathtt{ext}$ *or* $\mathtt{sim}$ *with* $pp := (pp, ck)$*, and any PPT algorithm* $\mathbf{A}$*,*

$$\begin{aligned}\Pr[&\mathsf{Prv.V}(x, c, \pi) = 1 \text{ if } (ck, x, w) \in R \mid \\ &(x, w) \leftarrow \mathbf{A}(pp), (c, r) \leftarrow \mathsf{Cmt.Com}(w), \\ &\pi \leftarrow \mathsf{Prv.P}(x, c, w, r)] = 1.\end{aligned}$$

Definition 7 (Perfect Soundness[18]) *A commit-and-prove scheme* CmtPrv *is said to be perfectly sound if it satisfies the following condition for some unbounded algorithm* $\mathsf{Cmt.Open}$*: For any security parameter* 1^λ*, any set of public parameters* $pp \leftarrow \mathsf{CmtPrv.Setup}(1^\lambda)$*, any commitment key* $ck \leftarrow \mathsf{Cmt.KG}(\mathtt{nor})$ *and any PPT algorithm* $\mathbf{A}$*,*

$$\begin{aligned}\Pr[&\mathsf{Prv.V}(x, c, \pi) = 0 \text{ or } (ck, x, w) \in R \mid \\ &(x, c, \pi) \leftarrow \mathbf{A}(pp), w \leftarrow \mathsf{Cmt.Open}(c)] = 1.\end{aligned}$$

Let C_{ck} be the set of commitments under ck to some message w.

Definition 8 (Perfect Knowledge Extraction[18]) *A commit-and-prove scheme* CmtPrv *is said to be perfectly knowledge extractable if it satisfies the following condition for some PPT algorithm* $\mathsf{Cmt.Ext}$*: For any security parameter* 1^λ*, any set of public parameters* $pp \leftarrow \mathsf{CmtPrv.Setup}(1^\lambda)$*, any commitment key* $(ck, xk) \leftarrow \mathsf{Cmt.KG}(\mathtt{ext})$ *and any PPT algorithm* $\mathbf{A}$*,*

$$\Pr[c \notin C_{ck} \text{ or } \mathsf{Cmt.Ext}(xk, c) = \mathsf{Cmt.Open}(c) \mid c \leftarrow \mathbf{A}(pp, ck, xk)] = 1.$$

Definition 9 (Composable Witness-Indistinguishability[18]) *A commit-and-prove scheme* CmtPrv *is said to be composably witness-indistinguishable if it satisfies the following condition: For any security parameter* 1^λ*, any set of public parameters* $pp \leftarrow \mathsf{CmtPrv.Setup}(1^\lambda)$ *and any PPT algorithm* $\mathbf{A}$*, the following holds.*

$$\begin{aligned}&\Pr[(ck, x, w), (ck, x, w') \in R \text{ and } \mathbf{A}(St, \pi) = 1 \mid (ck, tk) \leftarrow \mathsf{Cmt.KG}(\mathtt{sim}), pp := (pp, ck), \\ &\quad (x, w, w', St) \leftarrow \mathbf{A}^{\mathsf{Cmt.Com}(\cdot)}(pp, ck, tk), (c, r) \leftarrow \mathsf{Cmt.Com}(w), \pi \leftarrow \mathsf{Prv.P}(x, c, w, r)] \\ &= \Pr[(ck, x, w), (ck, x, w') \in R \text{ and } \mathbf{A}(St, \pi') = 1 \mid (ck, tk) \leftarrow \mathsf{Cmt.KG}(\mathtt{sim}), pp := (pp, ck), \\ &\quad (x, w, w', St) \leftarrow \mathbf{A}^{\mathsf{Cmt.Com}(\cdot)}(pp, ck, tk), (c', r') \leftarrow \mathsf{Cmt.Com}(w'), \pi' \leftarrow \mathsf{Prv.P}(x, c', w', r')]. \qquad (17)\end{aligned}$$

Especially, perfect witness-indistinguishability holds from Equation (17).

BUNDLED LANGUAGE

In this section, we define a notion of a bundled language in the case of a group-dependent language that is pairing-product equations. Intuitively, the notion is a *simultaneous* equation system whose coefficients form a language.

For a polynomially bounded integer q, we first prepare for q independent copies of an equation system with variables $(\hat{X}_i^a)_i$ and $(\check{Y}_j^a)_j$, as follows. (We remark that ^{a} is an index.)

$$\begin{aligned}&\text{For } a \in [q], \\ &\begin{cases} \sum_i \hat{X}_i^a \cdot \check{B}_{1i}^a + \sum_j \hat{A}_{1j}^a \cdot \check{Y}_j^a + \sum_i \sum_j \hat{X}_i^a \gamma_{1ij}^a \check{Y}_j^a = t_{\mathbb{T}1}^a, \\ \vdots \\ \sum_i \hat{X}_i^a \cdot \check{B}_{ni}^a + \sum_j \hat{A}_{nj}^a \cdot \check{Y}_j^a + \sum_i \sum_j \hat{X}_i^a \cdot \check{Y}_j^a = t_{\mathbb{T}n}^a. \end{cases} \qquad (18)\end{aligned}$$

Now we impose a constraint that the above q equation systems have a common variable. For simplicity, we enforce that

$$\hat{X}_1^1 = \cdots = \hat{X}_1^q = \hat{X}_1. \tag{19}$$

Definition 10 (Bundled language) *Let L be the language [Equation (10)]. For a polynomially bounded integer q, put $A := [q]$. The q-bundled language $\prod_{a\in A}^{bnd} L$ of the languages L is the subset of the q-Cartesian product of L with the constraint [Equation (19)]:*

$$\prod_{a\in A}^{bnd} L \stackrel{def}{=} \{(x^a)^{a\in A} \in \prod_{a\in A} L \mid \hat{X}_1^1 = \cdots = \hat{X}_1^q = \hat{X}_1\}. \tag{20}$$

DECENTRALIZED MULTI-AUTHORITY ANONYMOUS CREDENTIAL SYSTEM

In this section, we provide syntax and security definitions of dACS. We introduce three security definitions. The first is EUF against collusion attacks that cause mis-authentication, the other two are anonymity and unlinkability of proofs.

Syntax

Our dACS consists of five PPT algorithms, (Setup, AuthKG, PrivKG, Prover, Verifier). Intuitively, Setup generates a set of common public parameters pp. In a real use, it is executed only once by an entity that maintains a standard like NIST FIPS 186-4[16]. Then a key-issuing authority of an attribute executes AuthKG to generate its master secret key and public key. Then, the authority, being asked by an entity possessing an attribute, generates and issues a private secret key of the attribute for the entity. By using the private secret key(s), the entity executes Prover to generate a proof of knowing the key(s) of attribute(s). Receiving the proof, any verifier executes Verifier and decides whether the proof is valid to confirm the knowledge of attribute(s) or not.

• $\mathsf{Setup}(1^\lambda) \to pp$. This PPT algorithm is needed to generate a set of public parameters pp. On input the security parameter 1^λ, it generates the set pp. It returns pp.

• $\mathsf{AuthKG}(a) \to (\mathrm{PK}^a, \mathrm{MSK}^a)$. This PPT algorithm is executed by a key-issuing authority indexed by a. On input the authority index a, it generates the a-th public key PK^a of the authority and the corresponding a-th master secret key MSK^a. It returns $(\mathrm{PK}^a, \mathrm{MSK}^a)$.

• $\mathsf{PrivKG}(\mathrm{PK}^a, \mathrm{MSK}^a, \mathtt{i}) \to \mathrm{sk}_{\mathtt{i}}^a$. This PPT algorithm is executed by the a-th key-issuing authority. On input the a-th public and master secret keys $(\mathrm{PK}^a, \mathrm{MSK}^a)$ and an element $\mathtt{i} \in \hat{\mathbb{G}}$ (that is an identifier of a prover), it generates a private secret key $\mathrm{sk}_{\mathtt{i}}^a$ of a prover. It returns $\mathrm{sk}_{\mathtt{i}}^a$.

• $\mathsf{Prover}((\mathrm{PK}^a)^{a\in A'}, \mathtt{i}, (\mathrm{sk}_{\mathtt{i}}^a)^{a\in A'}) \to \pi$. This PPT algorithm is executed by a prover who is to be authenticated, where A' denotes a subset of the set A. On input the public keys $(\mathrm{PK}^a)^{a\in A'}$, an identity element $\mathtt{i}$ and the corresponding private secret keys $(\mathrm{sk}_{\mathtt{i}}^a)^{a\in A'}$, it returns a proof π.

• $\mathsf{Verifier}((\mathrm{PK}^a)^{a\in A'}, \pi) \to d$. This deterministic polynomial-time algorithm is executed by a verifier who confirms that the prover certainly knows the secret keys for indices $a \in A'$. On input the public keys $(\mathrm{PK}^a)^{a\in A'}$ and the proof π, it returns $d := 1$ ("accept") or $d := 0$ ("reject").

Security definitions

We define three security notions for our ACS dACS; EUF against collusion attacks, anonymity and unlinkability of proofs. Hereafter, the notation "(algorithm name)+**O**" means the oracle that functions as the algorithm.

EUF against collusion attack

Formally, we define the following experiment on dACS and an adversary algorithm **A**.

$$
\begin{aligned}
&\mathsf{Exp}^{\text{euf-coll}}_{\mathsf{dACS},\mathbf{A}}(1^\lambda, 1^\mu):\\
&pp \leftarrow \mathsf{Setup}(1^\lambda), A := [\mu], \text{For } a \in A: (\mathrm{PK}^a, \mathrm{MSK}^a) \leftarrow \mathsf{AuthKG}(a)\\
&(\tilde{A}, St) \leftarrow \mathbf{A}(pp, (\mathrm{PK}^a)^{a\in A}), \bar{\tilde{A}} := A \backslash \tilde{A}\\
&(\pi^*, A^*) \leftarrow \mathbf{A}^{\mathbf{PrivKO}(\mathrm{PK}^{\cdot}, \mathrm{MSK}^{\cdot}, \cdot)}(St, (\mathrm{MSK}^a)^{a\in\tilde{A}})\\
&\mathsf{Verifier}((\mathrm{PK}^a)^{a\in A^*}, \pi^*) \to d\\
&\text{If } d = 1 \text{ then return Win else return Lose}
\end{aligned}
$$

Intuitively, the above experiment describes the attack as follows. On input the public keys $(\mathrm{PK}^a)^{a\in A}$, **A** outputs a set $\tilde{A}$ of indices of corrupted authorities. Then **A** issues a query of the form $(a, \mathtt{i}_j)$, where $a \in \bar{\tilde{A}} := A\backslash\tilde{A}$ and $\mathtt{i}_j \in \hat{\mathbb{G}}$ for $j \in [q_{\mathrm{sk}}]$ to the private secret key oracle $\mathbf{PrivKO}(\mathrm{PK}^{\cdot}, \mathrm{MSK}^{\cdot}, \cdot)$. We denote by A_j the set of authority indices for which the private secret key queries were issued with $\mathtt{i}_j$. That is, $A_j := \{a \in A \mid \mathbf{A} \text{ is given } \mathrm{sk}^a_{\mathtt{i}_j}\} \subset \bar{\tilde{A}}$. We note that the maximum number of private secret key queries is $\mu \cdot q_{\mathrm{sk}}$. We require that the numbers μ and q_{sk} are bounded by a polynomial in λ. At the end **A** returns a forgery proof π^* together with the target set of authority indices A^* that is a subset of $\bar{\tilde{A}}$: $A^* \subset \bar{\tilde{A}}$. If the decision d on π^* by $\mathsf{Verifier}$ is 1 under $(\mathrm{PK}^a)^{a\in A^*}$, then the experiment returns Win; otherwise, it returns Lose.

A restriction is imposed on the adversary **A**: the queried $\mathtt{i}_j$s are pairwise different, and any A_j is a proper subset of the target set A^*:

$$\mathtt{i}_{j_1} \neq \mathtt{i}_{j_2} \text{ for } j_1, j_2 \in [q_{\mathrm{sk}}], j_1 \neq j_2, \tag{21}$$

$$A_j \subsetneq A^*, \ j \in [q_{\mathrm{sk}}]. \tag{22}$$

These restrictions are because, otherwise, the adversary **A** can trivially succeed in causing forgery.

The advantage of an adversary **A** over an ACS dACS in the experiment is defined as: $\mathbf{Adv}^{\text{euf-coll}}_{\mathsf{dACS},\mathbf{A}}(\lambda, \mu) \stackrel{\text{def}}{=} \Pr[\mathsf{Exp}^{\text{euf-coll}}_{\mathsf{dACS},\mathbf{A}}(1^\lambda, 1^\mu) = \text{Win}]$.

Definition 11 *A scheme dACS is said to be existentially unforgeable against collusion attacks if, for any PPT algorithm* **A**, *the advantage* $\mathbf{Adv}^{\textit{euf-coll}}_{\mathit{dACS},\mathbf{A}}(\lambda, \mu)$ *is negligible in* λ.

Anonymity of proofs

Formally, we define the following experiment on dACS and an adversary algorithm **A**.

$$
\begin{aligned}
&\mathsf{Exp}^{\text{ano-prf}}_{\mathsf{dACS},\mathbf{A}}(1^\lambda, 1^\mu):\\
&pp \leftarrow \mathsf{Setup}(1^\lambda), A := [\mu], \text{For } a \in A: (\mathrm{PK}^a, \mathrm{MSK}^a) \leftarrow \mathsf{AuthKG}(a)\\
&(\mathtt{i}_0, \mathtt{i}_1, St) \leftarrow \mathbf{A}(pp, (\mathrm{PK}^a)^{a\in A})\\
&\text{For } a \in A: \text{For } i = 0, 1: \mathrm{sk}^a_{\mathtt{i}_i} \leftarrow \mathsf{PrivKG}(\mathrm{PK}^a, \mathrm{MSK}^a, \mathtt{i}_i)\\
&b \in_R \{0, 1\}, b' \leftarrow \mathbf{A}^{\mathsf{Prover}((\mathrm{PK}^a)^{a\in A}, \mathtt{i}_b, (\mathrm{sk}^a_{\mathtt{i}_b})^{a\in A})}(St, (\mathrm{MSK}^a, \mathrm{sk}^a_{\mathtt{i}_0}, \mathrm{sk}^a_{\mathtt{i}_1})^{a\in A})\\
&\text{If } b = b' \text{ then return Win, else return Lose}
\end{aligned}
$$

Intuitively, the above experiment describes the attack as follows. On input the set of public parameters pp and the issued public keys $(\mathrm{PK}^a)^{a\in A}$, **A** designates two identity elements $\mathtt{i}_0$ and $\mathtt{i}_1$, and **A** is given two kinds of private secret keys $(\mathrm{sk}^a_{\mathtt{i}_0}, \mathrm{sk}^a_{\mathtt{i}_1})$ for all $a \in A$. Next, for randomly chosen $b \in \{0, 1\}$, which is hidden from **A**, **A** does oracle-access to a prover Prover that is on input the identity $\mathtt{i}_b$ and the private secret keys $(\mathrm{sk}^a_{\mathtt{i}_b})^{a\in A}$. If the decision b' of **A** is equal to b, then the experiment returns Win; otherwise, it returns Lose.

The advantage of an adversary **A** over an ACS dACS in the experiment is defined as: $\mathbf{Adv}^{\text{ano-prf}}_{\text{dACS},\mathbf{A}}(\lambda,\mu) \stackrel{\text{def}}{=} \left|\Pr[\mathsf{Exp}^{\text{ano-prf}}_{\text{dACS},\mathbf{A}}(1^\lambda,1^\mu)=\text{Win}]-(1/2)\right|$.

Definition 12 *An ACS dACS is said to have anonymity of proofs if, for any PPT algorithm* **A***, the advantage* $\mathbf{Adv}^{\textit{ano-prf}}_{\textit{dACS},\mathbf{A}}(\lambda,\mu)$ *is negligible in* λ.

Unlinkability of proofs

Formally we define the following experiment on dACS and an adversary algorithm **A**.

$$
\begin{aligned}
&\mathsf{Exp}^{\text{unlink-prf}}_{\text{dACS},\mathbf{A}}(1^\lambda,1^\mu):\\
&pp \leftarrow \mathsf{Setup}(1^\lambda), A := [\mu], \text{For } a \in A : (\mathrm{PK}^a, \mathrm{MSK}^a) \leftarrow \mathsf{AuthKG}(a)\\
&(\mathtt{i}_0, \mathtt{i}_1, St) \leftarrow \mathbf{A}(pp, (\mathrm{PK}^a)^{a\in A})\\
&\text{For } a \in A : \text{For } i = 0,1 : \mathrm{sk}^a_{\mathtt{i}_i} \leftarrow \mathsf{PrivKG}(\mathrm{PK}^a, \mathrm{MSK}^a, \mathtt{i}_i)\\
&b \in_R \{0,1\}\\
&\text{If } b = 0 \text{ then } St \leftarrow \mathbf{A}^{\mathsf{Prover}((\mathrm{PK}^a)^{a\in A}, \mathtt{i}_0, (\mathrm{sk}^a_{\mathtt{i}_0})^{a\in A})}(St, (\mathrm{MSK}^a, \mathrm{sk}^a_{\mathtt{i}_0}, \mathrm{sk}^a_{\mathtt{i}_1})^{a\in A})\\
&\qquad\qquad d \leftarrow \mathbf{A}^{\mathsf{Prover}((\mathrm{PK}^a)^{a\in A}, \mathtt{i}_1, (\mathrm{sk}^a_{\mathtt{i}_1})^{a\in A})}(St)\\
&\text{else} \qquad\quad St \leftarrow \mathbf{A}^{\mathsf{Prover}((\mathrm{PK}^a)^{a\in A}, \mathtt{i}_0, (\mathrm{sk}^a_{\mathtt{i}_0})^{a\in A})}(St, (\mathrm{MSK}^a, \mathrm{sk}^a_{\mathtt{i}_0}, \mathrm{sk}^a_{\mathtt{i}_1})^{a\in A})\\
&\qquad\qquad d \leftarrow \mathbf{A}^{\mathsf{Prover}((\mathrm{PK}^a)^{a\in A}, \mathtt{i}_0, (\mathrm{sk}^a_{\mathtt{i}_0})^{a\in A})}(St)\\
&\text{If } b = d \text{ then return Win, else return Lose}
\end{aligned}
$$

Intuitively, the above experiment resembles the experiment of anonymity $\mathsf{Exp}^{\text{ano-prf}}_{\text{dACS},\mathbf{A}}(1^\lambda,1^\mu)$. The difference is that, in the above experiment, the adversary **A** has to distinguish whether the proofs (π) are of the same entity or of the other entity.

The advantage of an adversary **A** over an ACS dACS in the experiment is defined as: $\mathbf{Adv}^{\text{unlink-prf}}_{\text{dACS},\mathbf{A}}(\lambda,\mu) \stackrel{\text{def}}{=} \left|\Pr[\mathsf{Exp}^{\text{unlink-prf}}_{\text{dACS},\mathbf{A}}(1^\lambda,1^\mu)=\text{Win}]-(1/2)\right|$.

Definition 13 *An ACS dACS is said to have unlinkability of proofs if, for any PPT algorithm* **A***, the advantage* $\mathbf{Adv}^{\textit{unlink-prf}}_{\textit{dACS},\mathbf{A}}(\lambda,\mu)$ *is negligible in* λ.

Proposition 1 (Unlinkability Implies Anonymity) *For any PPT algorithm* **A** *that is in accordance with the experiment* $\mathsf{Exp}^{\textit{ano-prf}}_{\textit{dACS},\mathbf{A}}(1^\lambda,1^\mu)$*, there exists a PPT algorithm* **B** *that is in accordance with the experiment* $\mathsf{Exp}^{\textit{unlink-prf}}_{\textit{dACS},\mathbf{B}}(1^\lambda,1^\mu)$ *and the following inequality holds.*

$$\mathbf{Adv}^{\textit{ano-prf}}_{\textit{dACS},\mathbf{A}}(\lambda,\mu) \le \mathbf{Adv}^{\textit{unlink-prf}}_{\textit{dACS},\mathbf{B}}(\lambda,\mu).$$

Proof. Suppose that any PPT algorithm **A** that is in accordance with the experiment $\mathsf{Exp}^{\text{ano-prf}}_{\text{dACS},\mathbf{A}}(1^\lambda,1^\mu)$ is given. Then we construct a PPT algorithm **A** that is in accordance with the experiment $\mathsf{Exp}^{\text{unlink-prf}}_{\text{dACS},\mathbf{B}}(1^\lambda,1^\mu)$ as follows. **B** employs A as a subroutine. **B** is able to generate **A**'s input by using **B**'s input and **A**'s output. Also, **B** is able to answer to **A**'s queries by issuing queries to **B**'s oracle and using the answers. Finally, when **A** outputs b', **B** sets $d := b'$. □

GENERIC CONSTRUCTION

In this section, we provide a generic construction of the scheme dACS. Here we employ two building blocks. One is the structure-preserving signature scheme[17,35]. The other is the commit-and-prove scheme of the

"fine-tuning Groth-Sahai proofs" system [18,19] on pairing-product equations of our "bundled language".

Construction

According to our syntax, the scheme dACS consists of five PPT algorithms: $\mathsf{dACS} = (\mathsf{Setup}, \mathsf{AuthKG}, \mathsf{PrivKG}, \mathsf{Prover}, \mathsf{Verifier})$.

• $\mathsf{Setup}(1^\lambda) \to pp$. On input the security parameter 1^λ, it runs the generation algorithm of bilinear groups, and it sets the output as a set of public parameters: $\mathcal{BG}(1^\lambda) \to (p, \hat{\mathbb{G}}, \check{\mathbb{G}}, \mathbb{T}, e, \hat{G}, \check{G}) =: pp$. Note that pp is common for both the structure-preserving signature scheme Sig and the commit-and-prove scheme CmtPrv. Besides, it runs the generation algorithm of commitment key: $\mathsf{Cmt.KG}(\mathtt{nor}) \to ck$. It returns $pp := (pp, ck)$.

• $\mathsf{AuthKG}(a) \to (\mathrm{PK}^a, \mathrm{MSK}^a)$. On input an authority index a, it executes the key-generation algorithm $\mathsf{Sig.KG}()$ to obtain $(\mathrm{PK}, \mathrm{SK})$. It sets $\mathrm{PK}^a := \mathrm{PK}$ and $\mathrm{MSK}^a := \mathrm{SK}$. It returns $(\mathrm{PK}^a, \mathrm{MSK}^a)$.

• $\mathsf{PrivKG}(\mathrm{PK}^a, \mathrm{MSK}^a, \mathtt{i}) \to \mathrm{sk}^a_{\mathtt{i}}$. On input PK^a, MSK^a and an element $\mathtt{i} \in \hat{\mathbb{G}}$, it sets $\mathrm{PK} := \mathrm{PK}^a$ and $\mathrm{SK} := \mathrm{MSK}^a$ and $m := \hat{M} := \mathtt{i}$. It executes the signing algorithm $\mathsf{Sig.Sign}(\mathrm{PK}, \mathrm{SK}, m)$ to obtain a signature σ. It sets $\mathrm{sk}^a_{\mathtt{i}} := \sigma$. It returns $\mathrm{sk}^a_{\mathtt{i}}$.

• $\mathsf{Prover}((\mathrm{PK}^a)^{a\in A'}, \mathtt{i}, (\mathrm{sk}^a_{\mathtt{i}})^{a\in A'}) \to \pi$. On input $(\mathrm{PK}^a)^{a\in A'}$, $\mathtt{i}$ and $(\mathrm{sk}^a_{\mathtt{i}})^{a\in A'}$, it first commits to $\mathtt{i}$:

$$c_0 \leftarrow \mathsf{Cmt.Com}(\mathtt{i}; r_0).$$

Second, for each $a \in A'$, it commits to the components $(\sigma^a_k)_k$ of the signature σ^a in the componentwise way.

$$(c^a_k)_k \leftarrow \mathsf{Cmt.Com}((\sigma^a_k)_k; (r^a_k)_k).$$

Then, for each authority index a it sets $x^a := \mathrm{PK}^a$. It also sets $c^a := (c_0, (c^a_k)_k)$, $w^a := (w_0, (w^a_k)_k) := (\mathtt{i}, (\sigma^a_k)_k)$ and $r^a := (r_0, (r^a_k)_k)$. It executes the prove-algorithm to obtain a proof:

$$\pi^a \leftarrow \mathsf{Prv.P}(x^a, c^a, w^a, r^a), a \in A'.$$

It sets $\bar{\pi}^a := ((c^a_k)_k, \pi^a)$ for each $a \in A'$, and it merges all the $\bar{\pi}^a$s and the commitment c_0 as $\pi := (c_0, (\bar{\pi}^a)^{a\in A'})$. It returns π.

• $\mathsf{Verifier}((\mathrm{PK}^a)^{a\in A'}, \pi) \to d$. On input $((\mathrm{PK}^a)^{a\in A'}, \pi)$, it sets $x^a := \mathrm{PK}^a$ and it sets $c^a := (c_0, (c^a_k))$ for each $a \in A'$. Then it executes the verify-algorithm for each $a \in A'$ to obtain the decisions:

$$d^a \leftarrow \mathsf{Prv.V}(x^a, c^a, \pi^a), a \in A'.$$

If all the decisions d^as are 1, then it returns $d := 1$; otherwise, it returns $d := 0$.

Security proofs

Theorem 1 (EUF against Collusion Attacks) *For any PPT algorithm* $\mathbf{A}$ *that is in accordance with the experiment* $\mathsf{Exp}^{euf\text{-}coll}_{dACS,\mathbf{A}}(1^\lambda, 1^\mu)$*, there exists a PPT algorithm* $\mathbf{F}$ *that is in accordance with the experiment* $\mathsf{Exp}^{euf\text{-}cma}_{Sig,\mathbf{F}}(1^\lambda)$ *and the following inequality holds.*

$$\mathbf{Adv}^{euf\text{-}coll}_{dACS,\mathbf{A}}(\lambda, \mu) = \mu \cdot \mathbf{Adv}^{euf\text{-}cma}_{Sig,\mathbf{F}}(\lambda).$$

Theorem 1 means that, if the structure-preserving signature scheme Sig is existentially unforgeable against adaptive chosen-message attacks, then our dACS is EUF against collusion attacks.

Proof. Given any PPT algorithm $\mathbf{A}$ that is in accordance with the experiment $\mathsf{Exp}^{\text{euf-coll}}_{\mathsf{dACS},\mathbf{A}}(1^\lambda, 1^\mu)$, we construct a PPT algorithm $\mathbf{F}$ that generates an existential forgery of Sig according to the experiment $\mathsf{Exp}^{\text{euf-cma}}_{\mathsf{Sig},\mathbf{F}}(1^\lambda)$. $\mathbf{F}$

is given as input the set of public parameters pp and a public key $\mathrm{PK}_{\mathsf{Sig}}$. **F** is also given an auxiliary input μ. **F** executes $\mathsf{Cmt.KG}(\mathtt{ext})$ to obtain a pair (ck, xk). **F** sets $pp := (pp, ck)$. **F** invokes the algorithm **A** with 1^λ to obtain the number μ and St. **F** chooses a *target index* $a^\dagger$ from the set $A := [\mu]$ *uniformly at random*. **F** executes the generation algorithm of an authority key honestly for $a \in A$ *except* the target index $a^\dagger$. As for $a^\dagger$, **F** uses the input public key:

$$\begin{aligned}&\text{For } a \in A, a \neq a^\dagger : (\mathrm{PK}^a, \mathrm{MSK}^a) \leftarrow \mathsf{AuthKG}(a),\\ &\text{For } a = a^\dagger : \mathrm{PK}^{a^\dagger} := \mathrm{PK}_{\mathsf{Sig}}.\end{aligned}$$

F inputs St and the public keys $(\mathrm{PK}^a)^{a\in A}$ into **A**. Then **F** obtains a set of corrupted authority indices $\tilde{A}$ from **A**. **F** sets $\bar{\tilde{A}} := A \backslash \tilde{A}$. If $a^\dagger \in \bar{\tilde{A}}$ (the case TgtIdx_1), then $a^\dagger$ is not in $\tilde{A}$ and **F** is able to input $[St, (\mathrm{MSK}^a)^{a\in\tilde{A}}]$ into **A**. Otherwise, **F** aborts.

Simulation of private secret key oracle. When **A** issues a private secret key query with $a \in A_j \subsetneq \bar{\tilde{A}}$ and $\mathtt{i}_j \in \mathbb{Z}_p (j = 1, \ldots, q_{\mathrm{sk}})$, **F** executes the generation algorithm of private secret key with $\mathtt{i}_j$ honestly for $a \in \bar{\tilde{A}}$ such that $a \neq a^\dagger$. As for $a = a^\dagger$, **F** issues a signing query to its oracle with $\mathtt{i}_j$:

$$\begin{aligned}&\text{For } a \in \bar{\tilde{A}} \text{ s.t. } a \neq a^\dagger : \mathrm{sk}^a_{\mathtt{i}_j} \leftarrow \mathsf{PrivKG}(\mathrm{PK}^a, \mathrm{MSK}^a, \mathtt{i}_j),\\ &\text{For } a = a^\dagger, \mathrm{sk}^{a^\dagger}_{\mathtt{i}_j} \leftarrow \mathbf{SignO}(\mathrm{PK}, \mathrm{SK}, \mathtt{i}_j).\end{aligned}$$

F replies to **A** with the secret key $\mathrm{sk}^a_{\mathtt{i}_j}$. This is a perfect simulation.

At the end **A** returns a forgery proof and the target set of authority indices (π^*, A^*). Note here that $A^* \subset \bar{\tilde{A}}$ as in the definition.

Generating Existential Forgery. Next, **F** runs a $\mathsf{Verifier}$ with an input $[(\mathrm{PK}^a)^{a\in A^*}, \pi^*]$. If the decision d of $\mathsf{Verifier}$ is 1, then **F** executes for each $a \in A^*$ the extraction algorithm $\mathsf{Cmt.Ext}(xk, c^a)$ to obtain a committed message $(w^a)^* = ((w^a_0)^*, ((w^a_k)^*)_k)$ (see Definition 8). Note here that, for all $a \in A^*$, $(w^a_0)^*$ is equal to a single element $(w_0)^*$ in $\hat{\mathbb{G}}$. This is because of the *perfectly binding property* of Cmt. Then **F** sets $\mathtt{i}^* := (w_0)^*$. Here the restriction [Equations (21)and (22)] assures that, if $q_{\mathrm{sk}} > 0$, then there exists at least one $\hat{a} \in (A^* \backslash A_j)$ for some $j \in [q_{\mathrm{sk}}]$. If $q_{\mathrm{sk}} = 0$, then there exists at least one $\hat{a} \in A^*$. **F** chooses one such $\hat{a}$ and sets $\sigma^* := (\sigma^{\hat{a}})^* := [(w^{\hat{a}}_k)^*]_k$. **F** returns a forgery pair of a message and a signature $(\mathtt{i}^*, \sigma^*)$. This completes the description of **F**.

Probability evaluation. The probability that the returned value $(\mathtt{i}^*, \sigma^*)$ is actually an existential forgery is evaluated as follows. We name the events in the above **F** as:

$$\begin{aligned}\mathrm{Acc} &: d = 1,\\ \mathrm{Ext} &: \mathsf{Cmt.Ext} \text{ returns a witness } (w^a)^*\\ \mathrm{TgtIdx} &: \hat{a} = a^\dagger,\\ \mathrm{Forge} &: (\mathtt{i}^*, \sigma^*) \text{ is an existential forgery on } \mathsf{Sig}.\end{aligned}$$

We have the following equalities.

$$\mathbf{Adv}^{\text{euf-coll}}_{\mathsf{dACS},\mathbf{A}}(\lambda, \mu) = \Pr[\mathrm{Acc}], \tag{23}$$

$$\Pr[\mathrm{Acc}, \mathrm{Ext}, \mathrm{TgtIdx}] = \Pr[\mathrm{Forge}], \tag{24}$$

$$\Pr[\mathrm{Forge}] = \mathbf{Adv}^{\text{euf-cma}}_{\mathsf{Sig},\mathbf{F}}(\lambda). \tag{25}$$

The left-hand side of the equality [Equation (24)] is expanded as follows.

$$\begin{aligned}\Pr[\mathrm{Acc}, \mathrm{Ext}, \mathrm{TgtIdx}] &= \Pr[\mathrm{TgtIdx}] \cdot \Pr[\mathrm{Acc}, \mathrm{Ext}] \\ &= \Pr[\mathrm{TgtIdx}] \cdot \Pr[\mathrm{Acc}] \cdot \Pr[\mathrm{Ext} \mid \mathrm{Acc}]. \end{aligned} \tag{26}$$

Claim 1

$$\Pr[\mathrm{TgtIdx}] = 1/|A| = 1/\mu. \tag{27}$$

Proof. $\hat{a}$ coincides with $a^\dagger$ with probability $1/|A|$ because $a^\dagger$ is chosen uniformly at random from A by **F** and no information of $a^\dagger$ is leaked to **A**. □

Claim 2 *If* TgtIdx *occurs, then* i^* *is not queried by* **F** *to its oracle* **SignO**.

Proof. This is because of the restriction [Equations (21) and (22)]. □

Claim 3

$$\Pr[\mathrm{Ext} \mid \mathrm{Acc}] = 1. \tag{28}$$

Proof. This is because of the perfect knowledge extraction of Π (see Definition 8). □

Combining Equations (23)-(28) we have:

$$\mathbf{Adv}^{\text{euf-coll}}_{\mathsf{dACS},\mathbf{A}}(\lambda, \mu) = \mu \cdot \mathbf{Adv}^{\text{euf-cma}}_{\mathsf{Sig},\mathbf{F}}(\lambda), \tag{29}$$

as is claimed in Theorem 1. □

Theorem 2 (Unlinkability of Proofs) *For any PPT algorithm* **A** *that is in accordance with the experiment* $\mathsf{Exp}^{\textit{unlink-prf}}_{\textit{dACS},\mathbf{A}}(1^\lambda, 1^\mu)$*, there exists a PPT algorithm* **D** *and the following inequality holds.*

$$\mathbf{Adv}^{\textit{unlink-prf}}_{\textit{dACS},\mathbf{A}}(\lambda, \mu) \leq \mathbf{Adv}^{\textit{ind-dual}}_{\textit{Cmt},\mathbf{D}}(\lambda).$$

[For the definition of $\mathbf{Adv}^{\textit{ind-dual}}_{\textit{Cmt},\mathbf{D}}(\lambda)$*, see Definition 3.]*

Theorem 2 means that, if the dual-mode commitment keys are indistinguishable, then our dACS has unlinkability.

Proof. Suppose that any PPT algorithm **A** that is in accordance with the experiment $\mathsf{Exp}^{\text{unlink-prf}}_{\mathsf{dACS},\mathbf{A}}(1^\lambda, 1^\mu)$ is given. We set a sequence of games, Game_0 and Game_1, as follows. Game_0 is exactly the same as $\mathsf{Exp}^{\text{unlink-prf}}_{\mathsf{dACS},\mathbf{A}}(1^\lambda, 1^\mu)$. Note that when a set of public parameters $pp = (pp', ck)$ is given to **A** where pp' is for bilinear groups, the commitment key ck is chosen as a commitment key ck of the mode `nor`. We denote the probability that Game_0 returns Win as $\Pr[\mathrm{Win}_0]$.

Game_1 is the same as Game_0 except that, when a set of public parameters $pp = (pp', ck)$ is given to **A**, the commitment key ck is chosen as a commitment key ck of the mode `sim`. We denote the probability that Game_1 returns Win as $\Pr[\mathrm{Win}_1]$. The values in Game_1 are distributed identically for both $\mathtt{i}_0$ and $\mathtt{i}_1$ due to the perfectly hiding property [Equation (16)] and the perfect witness-indistinguishability [Equation (17)]. Therefore, $\Pr[\mathrm{Win}_1] = 1/2$.

Employing **A** as a subroutine, we construct a PPT distinguisher algorithm **D** as follows. Given an input pp, ck, **D** reads out the security parameter. **D** simulates the environment of **A** in Game_0 or Game_1 honestly except

that $\mathbf{D}$ sets $pp := (pp, ck)$ instead of executing $\mathsf{Setup}(1^\lambda)$. If $b = b'$, then $\mathbf{D}$ returns 1, and otherwise, 0. By Equations (13) and (14) (see Definition 3), $\Pr[\mathbf{D}(pp, ck) = 1 \mid ck \leftarrow \mathsf{Cmt.KG}(\mathtt{nor})] = \Pr[\mathrm{Win}_0]$ and $\Pr[\mathbf{D}(pp, ck) = 1 \mid (ck, tk) \leftarrow \mathsf{Cmt.KG}(\mathtt{sim})] = \Pr[\mathrm{Win}_1]$, and

$$\mathbf{Adv}^{\text{ind-dual}}_{\mathsf{Cmt},\mathbf{D}}(\lambda) = |\Pr[\mathrm{Win}_0] - \Pr[\mathrm{Win}_1]|. \tag{30}$$

Therefore,

$$\begin{aligned}\mathbf{Adv}^{\text{unlink-prf}}_{\mathsf{dACS},\mathbf{A}}(\lambda, \mu) &= |\Pr[\mathrm{Win}_0] - (1/2)| \\ &\leq |\Pr[\mathrm{Win}_0] - \Pr[\mathrm{Win}_1]| + |\Pr[\mathrm{Win}_1] - (1/2)| \\ &= \mathbf{Adv}^{\text{ind-dual}}_{\mathsf{Cmt},\mathbf{D}}(\lambda) + 0 = \mathbf{Adv}^{\text{ind-dual}}_{\mathsf{Cmt},\mathbf{D}}(\lambda),\end{aligned} \tag{31}$$

as is claimed in Theorem 2. □

INSTANTIATION

In this section, we instantiate our dACS in bilinear groups of Type 3 pairing[20]. The security properties are guaranteed under the SXDH assumption[35,38]. In accordance with the generic construction in the previous section, we employ two building blocks. One is the structure-preserving signature scheme by Abe *et al.*[17], and the other is the commit-and-prove scheme of the "fine-tuning Groth-Sahai proofs" system by Escala-and-Groth[19] on the pairing-product equations of our bundled language, which is a simultaneous verification equation system of the structure-preserving signatures on an identity element $\mathtt{i}$.

Construction

According to our syntax, our instantiated scheme $\mathsf{dACS}_{\mathrm{sxdh}}$ consists of five PPT algorithms: $\mathsf{dACS}_{\mathrm{sxdh}} = (\mathsf{Setup}, \mathsf{AuthKG}, \mathsf{PrivKG}, \mathsf{Prover}, \mathsf{Verifier})$.

• $\mathsf{Setup}(1^\lambda) \to pp$. On input the security parameter 1^λ, it runs the generation algorithm of bilinear groups, and it sets the output as a set of public parameters: $\mathcal{BG}(1^\lambda) \to (p, \hat{\mathbb{G}}, \check{\mathbb{G}}, \mathbb{T}, e, \hat{G}, \check{G}) =: pp$. Note that pp is common for both the commit-and-prove scheme CmtPrv and the structure-preserving signature scheme Sig. Besides, it runs the generation algorithm of a commitment key: $\mathsf{Cmt.KG}(\mathtt{nor}) \to ck$. That is, it generates a CRS of mode $\mathtt{nor}$ for the Groth-Sahai NIZK proof system[19] as Diffie-Hellman tuples, as follows. It first samples $\chi, \xi, \chi', \xi' \in_R \mathbb{Z}_p$, and it computes $\hat{Q} := \chi\hat{G}, \hat{U} := \xi\hat{G}, \hat{V} := \chi\xi\hat{G}, \check{Q} := \chi'\check{G}, \check{U} := \xi'\check{G}, \check{V} := \chi'\xi'\check{G}$. Then it sets

$$\hat{\mathbf{crs}}_\Pi := \begin{bmatrix} \hat{G} & \hat{Q} \\ \hat{U} & \hat{V} \end{bmatrix}, \check{\mathbf{crs}}_\Pi := \begin{bmatrix} \check{G} & \check{Q} \\ \check{U} & \check{V}, \end{bmatrix}, \tag{32}$$

and it sets

$$ck := (\hat{\mathbf{crs}}_\Pi, \check{\mathbf{crs}}_\Pi).$$

It returns $pp := (pp, ck)$.

• $\mathsf{AuthKG}(a) \to (\mathrm{PK}^a, \mathrm{MSK}^a)$. On input an authority index a, it executes the key-generation algorithm of the structure-preserving signature scheme[17], $\mathsf{Sig.KG}(1^\lambda)$, to obtain (PK, SK). Precisely, it generates a CRS of mode $\mathtt{nor}$ for the Groth-Sahai proof systems[19] $\Pi_{\mathsf{Sig},0}$ and $\Pi_{\mathsf{Sig},1}$ as Diffie-Hellman tuples, as follows. (Note that the authority index a is omitted for simplicity). That is, for $i = 0, 1$, it first samples $\chi_i, \xi_i, \chi'_i, \xi'_i \in_R \mathbb{Z}_p$, and it computes $\hat{Q}_i := \chi_i\hat{G}, \hat{U}_i := \xi_i\hat{G}, \hat{V}_i := \chi_i\xi_i\hat{G}, \check{Q}_i := \chi'_i\check{G}, \check{U}_i := \xi'_i\check{G}, \check{V}_i := \chi'_i\xi'_i\check{G}$. Then it sets

$$\hat{\mathbf{crs}}_0 := \begin{bmatrix} \hat{G} & \hat{Q}_0 \\ \hat{U}_0 & \hat{V}_0 \end{bmatrix}, \hat{\mathbf{crs}}_1 := \begin{bmatrix} \hat{G} & \hat{Q}_1 \\ \hat{U}_1 & \hat{V}_1 \end{bmatrix}, \check{\mathbf{crs}}_1 := \begin{bmatrix} \check{G} & \check{Q}_1 \\ \check{U}_1 & \check{V}_1 \end{bmatrix}. \tag{33}$$

Then it generates exponent values as

$$x_0 \in_R \mathbb{Z}_p,\ x_1 := x_2 := 0. \tag{34}$$

Then it generates the secret keys of the ElGamal encryption as

$$y_0, y_1, y_2 \in_R \mathbb{Z}_p,\ \check{Y}_0 := y_0\check{G}, \check{Y}_1 := y_1\check{G}, \check{Y}_2 := y_2\check{G}. \tag{35}$$

Then it generates commitments as

$$\begin{aligned}
&[\hat{x_0}]_0 := \mathsf{Com}(\hat{\mathbf{crs}}_0, x_0; r_{x_{00}}),\ [\hat{x_1}]_0 := \mathsf{Com}(\hat{\mathbf{crs}}_0, x_1; r_{x_{10}}),\\
&[\check{x_2}]_1 := \mathsf{Com}(\check{\mathbf{crs}}_1, x_2; r_{x_{21}}), [\hat{y_0}]_0 := \mathsf{Com}(\hat{\mathbf{crs}}_0, y_0; r_{y_{00}}),\\
&[\hat{y_0}]_1 := \mathsf{Com}(\hat{\mathbf{crs}}_1, y_0; r_{y_{01}}), [\hat{y_1}]_1 := \mathsf{Com}(\hat{\mathbf{crs}}_1, y_1; r_{y_{11}}),\\
&[\check{y_2}]_1 := \mathsf{Com}(\check{\mathbf{crs}}_1, y_2; r_{y_{21}}),
\end{aligned} \tag{36}$$

where $[\hat{\mathsf{x}}]_i$ and $[\check{\mathsf{y}}]_j$ $(i = 0, 1,\ j = 0, 1)$ are computed as follows.

$$\begin{aligned}
[\hat{\mathsf{x}}]_i &:= \mathsf{Com}(\hat{\mathbf{crs}}_i, \mathsf{x}; r_{\mathsf{x}_i}) := (\mathsf{x}\hat{U}_i + r_{\mathsf{x}_i}\hat{G}, \mathsf{x}(\hat{V}_i + \hat{G}) + r_{\mathsf{x}_i}\hat{Q}_i),\\
[\check{\mathsf{y}}]_j &:= \mathsf{Com}(\check{\mathbf{crs}}_j, \mathsf{y}; r_{\mathsf{y}_i}) := (\mathsf{y}\check{U}_j + r_{\mathsf{y}_i}\check{G}, \mathsf{y}(\check{V}_j + \check{G}) + r_{\mathsf{y}_i}\check{Q}_j).
\end{aligned}$$

Then it generates a basis of messages as

$$\omega \in_R \mathbb{Z}_p,\ \check{G}_r := \omega\check{G},\ \gamma_1 \in_R \mathbb{Z}_p,\ \check{G}_1 := \gamma_1\check{G}_r. \tag{37}$$

Finally, it sets

$$\mathrm{PK} := (\hat{\mathbf{crs}}_0, \hat{\mathbf{crs}}_1, \check{\mathbf{crs}}_1, \check{Y}_0, \check{Y}_1, \check{Y}_2, [\hat{x_0}]_0, [\hat{x_1}]_0, [\check{x_2}]_1, [\hat{y_0}]_0, [\hat{y_0}]_1, [\hat{y_1}]_1, [\check{y_2}]_1, \check{G}_r, \check{G}_1), \tag{38}$$

$$\mathrm{SK} := (x_0, y_0, y_1, y_2, r_{x_{00}}, r_{x_{10}}, r_{x_{21}}, r_{y_{00}}, r_{y_{01}}, r_{y_{11}}, r_{y_{21}}, \omega, \gamma_1). \tag{39}$$

It sets $\mathrm{PK}^a := \mathrm{PK}$ and $\mathrm{MSK}^a := \mathrm{SK}$. It returns $(\mathrm{PK}^a, \mathrm{MSK}^a)$.

• $\mathsf{PrivKG}(\mathrm{PK}^a, \mathrm{MSK}^a, \mathtt{i}) \to \mathrm{sk}^a_{\mathtt{i}}$. On input PK^a, MSK^a and an element $\mathtt{i} \in \hat{\mathbb{G}}$, it sets $\mathrm{PK} := \mathrm{PK}^a$ and $\mathrm{SK} := \mathrm{MSK}^a$ and $m := \hat{M} := \mathtt{i}$. It executes the signing algorithm $\mathsf{Sig.Sign}(\mathrm{PK}, \mathrm{SK}, m)$ to obtain a signature σ on $\mathtt{i}$. Precisely, it generates a one-time key pair $\alpha \in_R \mathbb{Z}_p^*$ and $\check{A} := \alpha\check{G}$, and it generates a one-time signature $(\hat{Z}, \hat{R})$ as

$$\rho \in_R \mathbb{Z}_p,\ \hat{Z} := (\alpha - \rho\omega)\hat{G},\ \hat{R} := \rho\hat{G} - \gamma_1\hat{M}. \tag{40}$$

Then it encrypts $z_0 = z_1 := x_0$ and $z_2 := 0$ as

$$s \in_R \mathbb{Z}_p,\ (\check{E}_{z_0}, \check{E}_{z_1}, \check{E}_s) := (z_0\check{G} + s\check{Y}_0, z_1\check{G} + s\check{Y}_1, s\check{G}), \tag{41}$$

$$t \in_R \mathbb{Z}_p,\ (\hat{E}_{z_2}, \hat{E}_t) := (z_2\hat{G} + t\hat{Y}_2, t\hat{G}). \tag{42}$$

Then it generates commitments as

$$\begin{aligned}
&[\check{z_0}]_0 := \mathsf{Com}(\check{\mathbf{crs}}_0, z_0; r_{z_{00}}),\\
&[\check{z_0}]_1 := \mathsf{Com}(\check{\mathbf{crs}}_1, z_0; r_{z_{01}}),\\
&[\check{z_1}]_1 := \mathsf{Com}(\check{\mathbf{crs}}_1, z_1; r_{z_{11}}),
\end{aligned} \tag{43}$$

$$[\hat{z_2}]_1 := \mathsf{Com}(\hat{\mathbf{crs}}_1, z_2; r_{z_{21}}). \tag{44}$$

where $[\hat{z_2}]_1$, $[\check{z_0}]_0$, $[\check{z_0}]_1$ and $[\check{z_1}]_1$ are computed as follows.

$$\begin{aligned}
[\check{z_0}]_0 &:= \mathsf{Com}(\check{\mathbf{crs}}_0, z_0; r_{z_{00}}) = (z_0\check{U}_0 + r_{z_{00}}\check{G}, z_0(\check{V}_0 + \check{G}) + r_{z_{00}}\check{Q}_0),\\
[\check{z_0}]_1 &:= \mathsf{Com}(\check{\mathbf{crs}}_1, z_0; r_{z_{01}}) = (z_0\check{U}_1 + r_{z_{01}}\check{G}, z_0(\check{V}_1 + \check{G}) + r_{z_{01}}\check{Q}_1),\\
[\check{z_1}]_1 &:= \mathsf{Com}(\check{\mathbf{crs}}_1, z_1; r_{z_{11}}) = (z_1\check{U}_1 + r_{z_{11}}\check{G}, z_1(\check{V}_1 + \check{G}) + r_{z_{11}}\check{Q}_1),\\
[\hat{z_2}]_1 &:= \mathsf{Com}(\hat{\mathbf{crs}}_1, z_2; r_{z_{21}}) = (z_2\hat{U}_1 + r_{z_{21}}\hat{G}, z_2(\hat{V}_1 + \hat{G}) + r_{z_{21}}\hat{Q}_1).
\end{aligned} \tag{45}$$

Then it generates commitments as

$$\begin{aligned}
[\hat{1}]_0 &:= \mathsf{Com}(\hat{\mathbf{crs}}_0, 1; 0) = (\hat{U}_0, \hat{V}_0 + \hat{G}),\\
[\hat{1}]_1 &:= \mathsf{Com}(\hat{\mathbf{crs}}_1, 1; 0) = (\hat{U}_1, \hat{V}_1 + \hat{G}),\\
[\check{1}]_1 &:= \mathsf{Com}(\check{\mathbf{crs}}_1, 1; 0) = (\check{U}_1, \check{V}_1 + \check{G}).
\end{aligned} \tag{46}$$

Then it generates Groth-Sahai proofs as

$$\begin{aligned}
\rho_{0,0} &:= \pi_{0,0},\\
\rho_{0,1} &:= \pi_{0,1},\\
\rho_{1,0} &:= (\theta_{1,0,1}, \theta_{1,0,2}, \pi_{1,0,1}, \pi_{1,0,2}),\\
\rho_{1,1} &:= \pi_{1,1},\\
\rho_{1,2} &:= \pi_{1,2},\\
\rho_{1,3} &:= \pi_{1,3},
\end{aligned} \tag{47}$$

where $\pi_{0,0}$, $\pi_{0,1}$, $\theta_{1,0,1}$, $\theta_{1,0,2}$, $\pi_{1,0,1}$, $\pi_{1,0,2}$, $\pi_{1,1}$, $\pi_{1,2}$ and $\pi_{1,3}$ are computed as follows.

$$\begin{aligned}
\pi_{0,0} &:= r_{z_{00}}\check{G} - r_{x_{00}}\check{G} - r_{x_{10}}\check{A},\\
\pi_{0,1} &:= 0\check{E}_{z_0} - r_{z_{00}}\check{G} - r_{y_{00}}\check{E}_s,\\
\theta_{1,0,1} &:= (z_0(r_{x_{21}} - r_{z_{21}}) - z_1(r_{x_{21}} - r_{z_{21}}))\hat{U}_1 + ((x_2 - z_2)(z_0 - z_1) - \psi)\hat{G},\\
\theta_{1,0,2} &:= (z_0(r_{x_{21}} - r_{z_{21}}) - z_1(r_{x_{21}} - r_{z_{21}}))(\hat{V}_1 + \hat{G}) + ((x_2 - z_2)(z_0 - z_1) - \psi)\hat{Q}_1,\\
\pi_{1,0,1} &:= (x_2(r_{z_{01}} - r_{z_{11}}) - z_2(r_{z_{01}} - r_{z_{11}}))\check{U}_1 + \psi\check{G},\\
\pi_{1,0,2} &:= (x_2(r_{z_{01}} - r_{z_{11}}) - z_2(r_{z_{01}} - r_{z_{11}}))(\check{V}_1 + \check{G}) + \psi\check{Q}_1,\\
\pi_{1,1} &:= 0\check{E}_{z_0} - r_{z_{01}}\check{G} - r_{y_{01}}\check{E}_s,\\
\pi_{1,2} &:= 0\check{E}_{z_1} - r_{z_{11}}\check{G} - r_{y_{11}}\check{E}_s,\\
\pi_{1,3} &:= 0\hat{E}_{z_2} - r_{z_{21}}\hat{G} - r_{y_{21}}\hat{E}_t.
\end{aligned} \tag{48}$$

Then it sets

$$\sigma := (\check{A}, \hat{Z}, \hat{R}, \check{E}_{z_0}, \check{E}_{z_1}, \check{E}_s, \hat{E}_{z_2}, \hat{E}_t, [\check{z}_0]_0, [\check{z}_0]_1, [\check{z}_1]_1, [\hat{z}_2]_1, \rho_{0,0}, \rho_{0,1}, \rho_{1,0}, \rho_{1,1}, \rho_{1,2}, \rho_{1,3}). \tag{49}$$

It sets $\mathsf{sk}_{\mathtt{i}}^a := \sigma$. It returns $\mathsf{sk}_{\mathtt{i}}^a$.

• $\mathsf{Prover}((\mathrm{PK}^a)^{a\in A'}, \mathtt{i}, (\mathsf{sk}_{\mathtt{i}}^a)^{a\in A'}) \to \pi$. According to the previous work on the structure-preserving signatures[17], verification equations are given as an equation system below. In the equation system, we denote $(\hat{C}_{\mathsf{X},1}, \hat{C}_{\mathsf{X},2})_i = [\hat{\mathsf{X}}]_i$ and $(\check{D}_{\mathsf{Y},1}, \check{D}_{\mathsf{Y},2})_j = [\check{\mathsf{Y}}]_j$ for a variable X, Y and the indices i, j. Also note that we put an index ^{a} to distinguish each equation for $a \in A'$, while the identity element $\hat{M}(= m = \mathtt{i})$ is *simultaneous* for

those equations.

For $\forall a \in A'$ (where a is not an exponent but an index)

// To check the one-time key, message and one-time signature $(\check{A}^a, \hat{M}, (\hat{Z}^a, \hat{R}^a))$:

$$\hat{G} \cdot \check{A}^a = \hat{Z}^a \cdot \check{G} + \hat{R}^a \cdot \check{G}_r + \hat{M} \cdot \check{G}_1, \tag{50}$$

// To check the proof $\rho^a_{0,0} = \pi^a_{0,0}$:

$$\begin{aligned}
(\hat{C}^a_{z_0,1} - \hat{C}^a_{x_0,1}) \cdot \check{G} - \hat{C}^a_{x_1,1} \cdot \check{A}^a &= \hat{G} \cdot \pi^a_{0,0}, \\
(\hat{C}^a_{z_0,2} - \hat{C}^a_{x_0,2}) \cdot \check{G} - \hat{C}^a_{x_1,2} \cdot \check{A}^a &= \hat{Q}^a_0 \cdot \pi^a_{0,0},
\end{aligned} \tag{51}$$

// To check the proof $\rho^a_{0,1} = \pi^a_{0,1}$:

$$\begin{aligned}
\hat{C}^a_{z_0,1} \cdot \check{E}^a_{z_0} - \hat{C}^a_{x_0,1} \cdot \check{G} - \hat{C}^a_{x_1,1} \cdot \check{E}^a_s &= \hat{G} \cdot \pi^a_{0,1}, \\
\hat{C}^a_{z_0,2} \cdot \check{E}^a_{z_0} - \hat{C}^a_{x_0,2} \cdot \check{G} - \hat{C}^a_{x_1,2} \cdot \check{E}^a_s &= \hat{Q}^a_0 \cdot \pi^a_{0,1},
\end{aligned} \tag{52}$$

// To check the proof $\rho^a_{1,0} = (\pi^a_{1,0,1}, \pi^a_{1,0,2}, \theta^a_{1,0,1}, \theta^a_{1,0,2})$:

$$\begin{aligned}
(\hat{C}^a_{z_0,1} - \hat{C}^a_{z_1,1}) \cdot (\check{D}^a_{x_2,1} + \check{D}^a_{z_2,1}) &= \hat{G} \cdot \pi^a_{1,0,1} + \theta^a_{1,0,1} \cdot \check{G}, \\
(\hat{C}^a_{z_0,2} - \hat{C}^a_{z_1,2}) \cdot (\check{D}^a_{x_2,1} + \check{D}^a_{z_2,1}) &= \hat{Q}^a_1 \cdot \pi^a_{1,0,1} + \theta^a_{1,0,2} \cdot \check{G}, \\
(\hat{C}^a_{z_0,1} - \hat{C}^a_{z_1,1}) \cdot (\check{D}^a_{x_2,2} + \check{D}^a_{z_2,2}) &= \hat{G} \cdot \pi^a_{1,0,2} + \theta^a_{1,0,1} \cdot \check{Q}^a_1, \\
(\hat{C}^a_{z_0,2} - \hat{C}^a_{z_1,2}) \cdot (\check{D}^a_{x_2,2} + \check{D}^a_{z_2,2}) &= \hat{Q}^a_1 \cdot \pi^a_{1,0,2} + \theta^a_{1,0,2} \cdot \check{Q}^a_1,
\end{aligned} \tag{53}$$

// To check the proof $\rho^a_{1,1} = \pi^a_{1,1}$:

$$\begin{aligned}
\hat{C}^a_{z_0,1} \cdot \check{E}^a_{z_0} - \hat{C}^a_{x_0,1} \cdot \check{G} - \hat{C}^a_{x_1,1} \cdot \check{E}^a_s &= \hat{G} \cdot \pi^a_{1,1}, \\
\hat{C}^a_{z_0,2} \cdot \check{E}^a_{z_0} - \hat{C}^a_{x_0,2} \cdot \check{G} - \hat{C}^a_{x_1,2} \cdot \check{E}^a_s &= \hat{Q}^a_0 \cdot \pi^a_{1,1},
\end{aligned} \tag{54}$$

// To check the proof $\rho^a_{1,2} = \pi^a_{1,2}$:

$$\begin{aligned}
\hat{C}^a_{z_0,1} \cdot \check{E}^a_{z_1} - \hat{C}^a_{x_0,1} \cdot \check{G} - \hat{C}^a_{x_1,1} \cdot \check{E}^a_s &= \hat{G} \cdot \pi^a_{1,2}, \\
\hat{C}^a_{z_0,2} \cdot \check{E}^a_{z_1} - \hat{C}^a_{x_0,2} \cdot \check{G} - \hat{C}^a_{x_1,2} \cdot \check{E}^a_s &= \hat{Q}^a_0 \cdot \pi^a_{1,2},
\end{aligned} \tag{55}$$

// To check the proof $\rho^a_{1,3} = \pi^a_{1,3}$:

$$\begin{aligned}
\hat{E}^a_{z_2} \cdot \check{D}^a_{z_0,1} - \hat{G} \cdot \check{D}^a_{x_0,1} - \hat{E}^a_t \cdot \check{D}^a_{x_1,1} &= \pi^a_{1,3} \cdot \check{G}, \\
\hat{E}^a_{z_2} \cdot \check{D}^a_{z_0,2} - \hat{G} \cdot \check{D}^a_{x_0,2} - \hat{E}^a_t \cdot \check{D}^a_{x_1,2} &= \pi^a_{1,3} \cdot \check{Q}^a_1.
\end{aligned} \tag{56}$$

We note that *each* of the Equations (50)-(56) can be transformed into the following canonical form.

$$\hat{\boldsymbol{x}} = (\hat{x}_1, \ldots, \hat{x}_m),\ \check{\boldsymbol{y}} = (\check{y}_1, \ldots, \check{y}_n)^\top,\ \Gamma \in \mathbb{Z}_p^{m \times n}, \tag{57}$$

$$\hat{\boldsymbol{x}} \Gamma \check{\boldsymbol{y}} = 0_{\mathbb{T}}. \tag{58}$$

Then the Prover algorithm applies the Groth-Sahai NIZK proof system to Equation (58), as follows. First, by using $\mathbf{c\hat{r}s}_\Pi$ and $\mathbf{c\check{r}s}_\Pi$, set

$$\hat{\boldsymbol{v}} := (\hat{Q}, \hat{G})^\top,\ \hat{\boldsymbol{w}} := (\hat{V}, \hat{U})^\top,\ \ \check{\boldsymbol{v}} := (\check{Q}, \check{G}),\ \check{\boldsymbol{w}} := (\check{V}, \check{U}). \tag{59}$$

Also, set scalar vectors as

$$\boldsymbol{e} := (0, 1) \in \mathbb{Z}_p^2,\ \ \boldsymbol{r}_x, \boldsymbol{s}_x \in_R \mathbb{Z}_p^m,\ \ \boldsymbol{r}_y, \boldsymbol{s}_y \in_R \mathbb{Z}_p^n. \tag{60}$$

Then, generate commitments as follows.

$$\hat{\boldsymbol{c}} \leftarrow \boldsymbol{e}^\top \hat{\boldsymbol{x}} + \hat{\boldsymbol{v}} \boldsymbol{r}_x + \hat{\boldsymbol{w}} \boldsymbol{s}_x \in \hat{\mathbb{G}}^{2 \times m}, \tag{61}$$

$$\check{\boldsymbol{d}} \leftarrow \check{\boldsymbol{y}} \boldsymbol{e} + \boldsymbol{r}_y^\top \check{\boldsymbol{v}} + \boldsymbol{s}_y^\top \check{\boldsymbol{w}} \in \check{\mathbb{G}}^{n \times 2}. \tag{62}$$

We stress that, in the computation of the commitment $\hat{\boldsymbol{c}}$ for the equation Equation (50), the randomness to compute the commitment $\hat{\boldsymbol{c}}_0 (\in \hat{\mathbb{G}}^{2\times 1})$ to $\hat{M} (= \mathtt{i})$ is common over all $a \in A'$, and hence the randomness is sampled only once. This is why a single identity $\mathtt{i}^*$ is extracted in the security proof of unforgeability against collusion attacks.

Then, from Equations (58), (61) and (62), generate

$$\check{\boldsymbol{\pi}}'_{\hat{\gamma}} := \boldsymbol{r}_x \Gamma \check{\boldsymbol{d}} \in \check{\mathbb{G}}^{1\times 2}, \qquad \check{\boldsymbol{\pi}}'_{\hat{w}} := \boldsymbol{s}_x \Gamma \check{\boldsymbol{d}} \in \check{\mathbb{G}}^{1\times 2}, \tag{63}$$

$$\hat{\boldsymbol{\pi}}'_{\check{\gamma}} := (\hat{\boldsymbol{c}} - \hat{\boldsymbol{\gamma}} \boldsymbol{r}_x - \hat{\boldsymbol{w}} \boldsymbol{s}_x) \Gamma \boldsymbol{r}_y \in \hat{\mathbb{G}}^{2\times 1}, \quad \hat{\boldsymbol{\pi}}'_{\check{w}} := (\hat{\boldsymbol{c}} - \hat{\boldsymbol{\gamma}} \boldsymbol{r}_x - \hat{\boldsymbol{w}} \boldsymbol{s}_x) \Gamma \boldsymbol{s}_y \in \hat{\mathbb{G}}^{2\times 1}. \tag{64}$$

Finally, obtain a Groth-Sahai proof $(\check{\boldsymbol{\pi}}_{\hat{\gamma}}, \check{\boldsymbol{\pi}}_{\hat{w}}, \hat{\boldsymbol{\pi}}_{\check{\gamma}}, \hat{\boldsymbol{\pi}}_{\check{w}})$ by randomizing Equations (64) and (65), as follows.

$$\alpha, \beta, \gamma, \delta \in_R \mathbb{Z}_p,$$

$$\check{\boldsymbol{\pi}}_{\hat{\gamma}} := \check{\boldsymbol{\pi}}'_{\hat{\gamma}} + \alpha \check{\boldsymbol{\gamma}} + \beta \check{\boldsymbol{w}} \in \check{\mathbb{G}}^{1\times 2}, \quad \check{\boldsymbol{\pi}}_{\hat{w}} := \check{\boldsymbol{\pi}}'_{\hat{w}} + \gamma \check{\boldsymbol{\gamma}} + \delta \check{\boldsymbol{w}} \in \check{\mathbb{G}}^{1\times 2}, \tag{65}$$

$$\hat{\boldsymbol{\pi}}_{\check{\gamma}} := \hat{\boldsymbol{\pi}}'_{\check{\gamma}} - \hat{\boldsymbol{\gamma}} \alpha - \hat{\boldsymbol{w}} \beta \in \hat{\mathbb{G}}^{2\times 1}, \quad \hat{\boldsymbol{\pi}}_{\check{w}} := \hat{\boldsymbol{\pi}}'_{\check{w}} - \hat{\boldsymbol{\gamma}} \gamma - \hat{\boldsymbol{w}} \delta \in \hat{\mathbb{G}}^{2\times 1}. \tag{66}$$

To summarize, the Prover algorithm obtains Groth-Sahai proofs by computing $(\hat{\boldsymbol{c}}, \check{\boldsymbol{d}}, \check{\boldsymbol{\pi}}_{\hat{\gamma}}, \check{\boldsymbol{\pi}}_{\hat{w}}, \hat{\boldsymbol{\pi}}_{\check{\gamma}}, \hat{\boldsymbol{\pi}}_{\check{w}})$ for each of the fifteen equations in Equations (50)-(56). We distinguish them by double index (a, k), where $a \in A'$ and $k \in [15]$. Then the algorithm returns all the commitments and proofs as $\boldsymbol{\pi}$, but the commitment $\hat{\boldsymbol{c}}_0$ to $\mathtt{i} (= \hat{M})$ is especially placed at the first entry. That is,

$$\boldsymbol{\pi} := (\hat{\boldsymbol{c}}_0, (\hat{\boldsymbol{c}}^{a,k}, \check{\boldsymbol{d}}^{a,k}, \check{\boldsymbol{\pi}}^{a,k}_{\hat{\gamma}}, \check{\boldsymbol{\pi}}^{a,k}_{\hat{w}}, \hat{\boldsymbol{\pi}}^{a,k}_{\check{\gamma}}, \hat{\boldsymbol{\pi}}^{a,k}_{\check{w}})^{a\in A', k\in[15]}). \tag{67}$$

• $\mathsf{Verifier}((\mathrm{PK}^a)^{a\in A'}, \pi) \to d$. On input $((\mathrm{PK}^a)^{a\in A'}, \boldsymbol{\pi})$, this verification algorithm does the checks the following evaluation.

For $a \in A'$:

For $k \in [15]$:

$$d^{a,k} := (\hat{\boldsymbol{c}}^{a,k} \Gamma \check{\boldsymbol{d}}^{a,k} =_? \hat{\boldsymbol{\gamma}} \cdot \check{\boldsymbol{\pi}}^{a,k}_{\hat{\gamma}} + \hat{\boldsymbol{w}} \cdot \check{\boldsymbol{\pi}}^{a,k}_{\hat{w}} + \hat{\boldsymbol{\pi}}^{a,k}_{\check{\gamma}} \cdot \check{\boldsymbol{\gamma}} + \hat{\boldsymbol{\pi}}^{a,k}_{\check{w}} \cdot \check{\boldsymbol{w}}) \text{ (as four equations in } \mathbb{T}^{2\times 2}) \tag{68}$$

$d^a := \wedge_{k\in[15]} d^{a,k}$

Return $d := \wedge_{a\in A'} d^a$.

Theorem 3 (EUF against Collusion Attacks) *For any PPT algorithm* $\mathbf{A}$ *that is in accordance with the experiment* $\mathsf{Exp}^{euf\text{-}coll}_{dACS,\mathbf{A}}(1^\lambda, 1^\mu)$*, our instantiated* $dACS_{sxdh}$ *is EUF against collusion attacks under the SXDH assumption.*

Theorem 4 (Unlinkability of Proofs) *For any PPT algorithm* $\mathbf{A}$ *that is in accordance with the experiment* $\mathsf{Exp}^{unlink\text{-}prf}_{dACS,\mathbf{A}}(1^\lambda, 1^\mu)$*, our instantiated* $dACS_{sxdh}$ *is unlinkable under the SXDH assumption.*

FEATURE COMPARISON AND EFFICIENCY EVALUATION

In this section, we compare the features of our instantiated $\mathrm{dACS}_{\mathrm{sxdh}}$ with those of the previous abACS. Then we evaluate efficiency of our $\mathrm{dACS}_{\mathrm{sxdh}}$ by partial implementation and estimation.

In Table 1, "Multi-Authority" means whether the issuing function is decentralized multi-authority or not. "Collusion Resistance" means whether the system has collusion resistance or not. "Formula of Proof" means the type of boolean formulas associated with the proofs. "R.O.", "Std." and "Gen.Grp." mean that the security proof is in the random oracle model, the standard model and the generic group model, respectively. "all-AND", "CNF", and "monotone" mean all-AND, CNF and monotone formulas, respectively. "Unlinkability" means

Table 1. Feature comparison of our dACS$_{sxdh}$ with previous work

ACS/Feature	Multi-authority	formulas for proofs	Security model	Unlink-ability	Unforge-ability	Collusion resistance	Size of a proof
GGM14 [10]	✓	All-AND	R.O.	✓	SRSA & DL	✓	$O(1)$
CDHK15 [21]	✓	All-AND	Std.	✓	SXDH,J-RootDH, *etc.*	✓	$O(\|A'\|)$
SNBF17 [24]	-	Monotone	Std.	✓	t-co-DL,	-	$O(2^{\|A'\|})$
ON19 [25]	-	CNF	Std.	✓	DLIN,q-SFP,n-DHE	-	$O(1)$
FHS19 [26]	-	All-AND	Gen.Grp.	✓	t-co-DL,	-	$O(1)$
TG20 [8]	-	Monotone	Std.	✓	(co-)SDH	✓	$O(\|A'\|)$
CY22 [9]	-	Monotone	Std.	✓	(co-)SDH	✓	$O(\|A'\|)$
Our dACS$_{sxdh}$	✓	All-AND	Std.	✓	SXDH	✓	$O(\|A'\|)$

ACS: Anonymous credential system; dACS: decentralized multi-authority attribute-based anonymous credential system.

Table 2. Number of elements in $(\hat{\boldsymbol{c}}^{a,k}, \check{\boldsymbol{d}}^{a,k})$

k	1	2	3	4	$\cdots$	15
(m,n)	$(4,4)$	$(4,3)$	$(4,3)$	$(4,4)$	$\cdots$	$(4,4)$
#elem.$\in \hat{\mathbb{G}} \times \check{\mathbb{G}}$	$(8,8)$	$(8,6)$	$(8,6)$	$(8,8)$	$\cdots$	$(8,8)$

Table 3. Number of elements in $(\check{\boldsymbol{\pi}}_{\hat{\mathrm{v}}}^{a,k}, \check{\boldsymbol{\pi}}_{\hat{\mathrm{w}}}^{a,k}, \hat{\boldsymbol{\pi}}_{\check{\mathrm{v}}}^{a,k}, \hat{\boldsymbol{\pi}}_{\check{\mathrm{w}}}^{a,k})$

k	1	$\ldots$	15
#elem.$\in \check{\mathbb{G}} \times \check{\mathbb{G}} \times \hat{\mathbb{G}} \times \hat{\mathbb{G}}$	$(2,2,2,2)$	$\cdots$	$(2,2,2,2)$

Table 4. Total number of elements in $\boldsymbol{\pi} = (\hat{\boldsymbol{c}}^{a,k}, \check{\boldsymbol{d}}^{a,k}, \check{\boldsymbol{\pi}}_{\hat{\mathrm{v}}}^{a,k}, \check{\boldsymbol{\pi}}_{\hat{\mathrm{w}}}^{a,k}, \hat{\boldsymbol{\pi}}_{\check{\mathrm{v}}}^{a,k}, \hat{\boldsymbol{\pi}}_{\check{\mathrm{w}}}^{a,k})$

k	1	2	3	4	$\cdots$	15	Further total over all k
#elem.$\in \hat{\mathbb{G}} \times \check{\mathbb{G}}$	$(12,12)$	$(12,10)$	$(12,10)$	$(12,12)$	$\cdots$	$(12,12)$	$(180,176)$

whether unlinkability of proofs is assured or not. "Unforgeability" means the assumptions under which unforgeability is assured. "Size of a Proof" means asymptotic behavior of data length of a proof of credentials. dACS$_{sxdh}$ means our instantiated dACS under the SXDH assumption. For each "Unforgeability Assumption", see the cited references.

Along with Table 1, feature comparison is explained at "Related Recent Work" in Introduction. We add a few remarks about the three decentralized multi-authority abACS (dACSs) in Table 1. The dACS by Garman *et al.* has good features, and especially it shows asymptotic behavior of a constant size of a proof[10]. However, its security model is in the random oracle model. The security model of the dACS by Camenisch *et al.* and our dACS$_{sxdh}$ is the standard model[21]. Both of them have similar features, and especially they show the same asymptotic behavior of linear complexity in the number of proven attribute credentials, $|A'|$. The difference lies in their unforgeability assumptions. Actually, in addition to the SXDH assumption, the dACS by Camenisch *et al.* needs more assumptions of J-RootDH, n-BSDH, q-SDH, XDLIN, co-CDH, and DBP[21].

Then, we show a concrete evaluation about computational amount of our dACS$_{sxdh}$. The number of group elements in a proof $\boldsymbol{\pi}$ is estimated as follows. $\boldsymbol{\pi}$ consists of components Equation (67), where the indices a and k run in $|A'|$ and $[15]$, respectively. In Equation (67), the commitments $\hat{\boldsymbol{c}}^{a,k}$ and $\check{\boldsymbol{d}}^{a,k}$ consist of group elements in $\hat{\mathbb{G}}$ and $\check{\mathbb{G}}$, and the number of the elements is decided by m and n, respectively. Table 2 shows the number of the elements. As for the Groth-Sahai proofs $\check{\boldsymbol{\pi}}_{\hat{\mathrm{v}}}^{a,k}$, $\check{\boldsymbol{\pi}}_{\hat{\mathrm{w}}}^{a,k}$, $\hat{\boldsymbol{\pi}}_{\check{\mathrm{v}}}^{a,k}$ and $\hat{\boldsymbol{\pi}}_{\check{\mathrm{w}}}^{a,k}$, each of them consists of two group elements in $\hat{\mathbb{G}}$ and $\check{\mathbb{G}}$ [Table 3]. Therefore, as summarized in Table 4, the number of group elements in our proof $\boldsymbol{\pi}$ is $(180,176) \in (\hat{\mathbb{G}}, \check{\mathbb{G}})$ for $|A'| = 1$. As an example, when $|A'| = 2$, it is $(358,352)$, where we subtract two group elements $\in \hat{\mathbb{G}}$ because the commitment $\hat{\boldsymbol{c}}_0$ to $\hat{M}(=\mathtt{i})$ is reused. When we use the TEPLA pairing library[39] that is in the C language, an element in $\hat{\mathbb{G}}$ is 65 bytes and an element in $\check{\mathbb{G}}$ is 129 bytes. Therefore, the data length of $\boldsymbol{\pi}$ is 34k bytes for $|A'| = 1$ and 68k byte for $|A'| = 2$.

Further, we estimate the execution time of the Prover and Verifier algorithms. Using TEPLA[39] in C, we

Table 5. Time of scalar-multiplication and pairing

Curve	Computation	Average (ms)(*2)
ECBN254a	$\hat{G}\alpha$	0.30
	$\beta\check{G}$	1.60
	$\hat{X}\cdot\check{Y}$(*1)	2.74

(*1) $\hat{X}\cdot\check{Y}=e(\hat{X},\check{Y})$; (*2) Intel Core i7-7600U CPU of 2.80 GHz.

Table 6. Estimated time of Prvr and Vrfr: $|A'|=2$

Curve	Algorithm	Estimation (s)
ECBN254a	Prover	2.8
	Verifier	2.4

implement the scalar-multiplications in $\hat{\mathbb{G}}$ and $\check{\mathbb{G}}$ and the bilinear map $e:\hat{\mathbb{G}}\times\check{\mathbb{G}}\to\mathbb{T}$ because the execution time of Prover is mostly occupied by them. The elliptic curve used is CBN254a known as the BN curve with 128 bit intended security ($\lambda=128$), which is one of the pairing-friendly curves. As for our hardware, CPU of 2.80 GHz and DRAM of 4 GB are used. Table 5 shows the result. One execution of elliptic scalar-multiplication in $\hat{\mathbb{G}}$, that is, $\hat{G}\alpha$ for $\alpha\in_R\mathbb{Z}_p$, needs about 0.30 ms on average. one execution in $\check{\mathbb{G}}$, that is, $\beta\check{G}$ for $\beta\in_R\mathbb{Z}_p$, needs about 1.60 ms on average. On the other hand, one execution of the pairing e, that is, $\hat{X}\cdot\check{Y}(=e(\hat{X},\check{Y}))$ for random $\hat{X}$ and $\check{Y}$, needs about 2.74 ms on average. Then we count the number of computations of the scalar-multiplications and the bilinear map to generate a proof $\boldsymbol{\pi}$ in a similar way to count the number of group elements. By a naive counting, it turns out that Prover needs $(324, 934)$ scalar-multiplications in $(\hat{\mathbb{G}},\check{\mathbb{G}})$, and Verifier needs 120 scalar-multiplication in $\hat{\mathbb{G}}$ and 292 computations of the bilinear map e. As an example, (Prover, Verifier) need about $(1.4, 1.2)$ s for $|A'|=1$ and $(2.8, 2.4)$ s for $|A'|=2$ [Table 6].

CONCLUSION

We propose a multi-show decentralized multi-authority abACS, dACS. One of the features in the security definitions is that corruption of authorities is introduced. As for the construction of our dACS, an attribute authority who issues a private secret key to an entity only has to sign the entity's identifier. Then the entity generates a proof of knowing credentials according to the "commit-to-identifier" principle. The generic construction actually employs the structure-preserving signature scheme and the Groth-Sahai non-interactive proof system in asymmetric bilinear groups. There the principle turns into a bundled language, which is simultaneous equations on the identifier, for verification of the structure-preserving signatures. Actually the bundled language works for preventing collusion attacks.

A negative aspect of our dACS is that the proof size is linear in the number of attribute credentials involved in a proof. Therefore, a construction with smaller asymptotic behavior, hopefully of constant size, should be our future work.

DECLARATIONS

Acknowledgments

The author would like to express his sincere thanks to the three anonymous reviewers as well as the associated editor for their helpful comments from their technical and editorial viewpoints. Part of this work was presented at Innovative Security Solutions for Information Technology and Communications - 13th International Conference, SecITC 2020 (Bucharest, Romania, November 19-20, 2020, pp.71-90)[34], and the sole author agrees to submit and publish it in this new article.

Authors' contributions

The author contributed solely to the article.

Availability of data and materials
Not applicable.

Financial support and sponsorship
This work was supported by JSPS KAKENHI Grant Number JP23K11106.

Conflicts of interest
The author declared that there are no conflicts of interest.

Ethical approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

REFERENCES
1. ISO/IEC 11578:1996(en). Available from: https://www.iso.org/obp/ui/#iso:std:iso-iec:11578:ed-1:v1:en. [Last accessed on 6 Sep 2024]
2. Chaum D. Security without identification: transaction systems to make big brother obsolete. *Commun ACM* 1985;28:1030-44. DOI
3. Camenisch J, Lysyanskaya A. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann B, editor. Advances in Cryptology - EUROCRYPT 2001. Lecture notes in computer science. Berlin, Heidelberg: Springer; 2001. pp. 93-118. DOI
4. Camenisch J, Lysyanskaya A. Signature schemes and anonymous credentials from bilinear maps. In: Advances in Cryptology - CRYPTO 2004. Lecture notes in computer science. Berlin, Heidelberg: Springer; 2004. pp. 56-72. DOI
5. Camenisch J, GroßT. Efficient attributes for anonymous credentials. In: CCS'08: Proceedings of the 15th ACM Conference on Computer and Communications Security. New York, NY, USA: ACM; 2008. pp. 345-56. DOI
6. Sudarsono A, Nakanishi T, Funabiki N. Efficient proofs of attributes in pairing-based anonymous credential system. In: Fischer-Hübner S, Hopper N, editors. Privacy enhancing technologies. PETS 2011. Lecture notes in computer science. Berlin, Heidelberg: Springer; 2011. pp. 246-63. DOI
7. Brands SA. Rethinking public key infrastructures and digital certificates: building in privacy. 1st ed. Cambridge-London: MIT Press; 2000. Available from: http://www.credentica.com/the_mit_pressbook.html. [Last accessed on 6 Sep 2024]
8. Tan S, Groß T. MoniPoly - an expressive *q*-SDH-based anonymous attribute-based credential system. In: Moriai S, Wang H, editors. Advances in Cryptology - ASIACRYPT 2020. Lecture notes in computer science. Cham: Springer; 2020. pp. 498-526. DOI
9. Chan KY, Yuen TH. Attribute-based anonymous credential: optimization for single-use and multi-use. In: Beresford AR, Patra A, Bellini E, editors. Cryptology and network security. CANS 2022. Lecture notes in computer science. Cham: Springer; 2022. pp. 89-121. DOI
10. Garman C, Green M, Miers I. Decentralized anonymous credentials. Available from: https://www.ndss-symposium.org/ndss2014/decentralized-anonymous-credentials. [Last accessed on 6 Sep 2024]
11. Lewko A, Waters B. Decentralizing attribute-Based encryption. In: Paterson KG, editor. Advances in Cryptology - EUROCRYPT 2011. Lecture notes in computer science. Berlin, Heidelberg: Springer; 2011. pp. 568-88. DOI
12. Okamoto T, Takashima K. Decentralized attribute-based signatures. In: Kurosawa K, Hanaoka G, editors. Public-Key Cryptography - PKC 2013. Lecture notes in computer science. Berlin, Heidelberg: Springer; 2013. pp. 125-42. DOI
13. Sahai A, Waters B. Fuzzy identity-based encryption. In: Cramer R, editor. Advances in Cryptology - EUROCRYPT 2005. Lecture notes in computer science. Berlin, Heidelberg: Springer; 2005. pp. 457-73. DOI
14. Goyal V, Pandey O, Sahai A, Waters B. Attribute-based encryption for fine-grained access control of encrypted data. In: CCS'06: Proceedings of the 13th ACM Conference on Computer and Communications Security. New York, NY, USA: Association for Computing Machinery; 2006. pp. 89-98. DOI
15. Chase M, Chow SSM. Improving privacy and security in multi-authority attribute-based encryption. In: CCS'09: Proceedings of the 2009 ACM Conference on Computer and Communications Security. New York, NY, USA: Association for Computing Machinery; 2009. pp. 121-30. DOI
16. NIST. Digital signature standard (DSS). 2013. Available from: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf. [Last accessed on 6 Sep 2024]
17. Abe M, Hofheinz D, Nishimaki R, Ohkubo M, Pan J. Compact structure - preserving signatures with almost tight security. In: Katz J, Shacham H, editors. Advances in Cryptology - CRYPTO 2017. Lecture notes in computer science. Cham: Springer; 2017. pp. 548-80. DOI

18. Groth J, Sahai A. Efficient non-interactive proof systems for bilinear groups. In: EUROCRYPT'08: Proceedings of the Theory and Applications of Cryptographic Techniques 27th Annual International Conference on Advances in Cryptology. Berlin, Heidelberg: Springer-Verlag; 2008. pp. 415-32. Available from: http://dl.acm.org/citation.cfm?id=1788414.1788438. [Last accessed on 6 Sep 2024]
19. Escala A, Groth J. Fine-tuning groth-Sahai Proofs. In: Krawczyk H, editor. Public-Key Cryptography - PKC 2014. Lecture notes in computer science. Berlin: Springer; 2014. pp. 630-49. DOI
20. Galbraith SD, Paterson KG, Smart NP. Pairings for cryptographers. *Discret Appl Math* 2008;156:3113-21. DOI
21. Camenisch J, Dubovitskaya M, Haralambiev K, Kohlweiss M. Composable and modular anonymous credentials: definitions and practical constructions. In: Iwata T, Cheon J, editors. Advances in Cryptology - ASIACRYPT 2015. Lecture notes in computer science. Berlin: Springer; 2015. pp. 262-88. DOI
22. Canetti R. Universally composable security: a new paradigm for cryptographic protocols. In: Proceedings 42nd IEEE Symposium on Foundations of Computer Science; 2001 Oct 8-11; Newport Beach, CA, USA. IEEE; 2001. pp. 136-45. DOI
23. Canetti R. Universally composable security. *J ACM* 2020;67:1-94. DOI
24. Sadiah S, Nakanishi T, Begum N, Funabiki N. Accumulator for monotone formulas and its application to anonymous credential system. *J Inf Process* 2017;25:949-61. DOI
25. Okishima R, Nakanishi T. An anonymous credential system with constant-size attribute proofs for CNF formulas with negations. In: Attrapadung N, Yagi T, editors. Advances in information and computer security. IWSEC 2019. Lecture notes in computer science. Cham: Springer; 2019. pp. 89-106. DOI
26. Fuchsbauer G, Hanser C, Slamanig D. Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. *J Cryptology* 2019;32:498-546. DOI
27. Resisting replay attacks efficiently in a permissioned and privacy-preserving blockchain network. US 20170149819 A1, United States Patent and Trademark Office. Available from: https://patents.google.com/patent/US20170149819A1/en. [Last accessed on 10 Sep 2024]
28. Limited AGH. System and method for detecting replay attack. US 20200128043 A1, United States Patent and Trademark Office. Available from: https://patents.google.com/patent/US20200128043A1/en. [Last accessed on 10 Sep 2024]
29. Nakamoto S. Bitcoin: a peer-to-peer electronic cash system. 2009. Available from: http://www.bitcoin.org/bitcoin.pdf. [Last accessed on 6 Sep 2024]
30. Au MH, Susilo W, Mu Y, Chow SSM. Constant-size dynamic K-times anonymous authentication. *IEEE Syst J* 2013;7:249-61. DOI
31. Ma JPK, Chow SSM. SMART credentials in the multi-queue of slackness (or secure management of anonymous reputation traits without global halting). In: 2023 IEEE 8th European Symposium on Security and Privacy EuroS&P; 2023 Jul 3-7; Delft, Netherlands. IEEE; 2023. pp. 896-912. DOI
32. Doerner J, Kondi Y, Lee E, Shelat A, Tyner L. Threshold BBS+ signatures for distributed anonymous credential issuance. In: 2023 IEEE Symposium on Security and Privacy SP; 2023 May 21-25; San Francisco, CA, USA. IEEE; 2023. pp. 773-89. DOI
33. Wong HWH, Ma JPK, Chow SSM. Secure multiparty computation of threshold signatures made more efficient. Available from: https://www.ndss-symposium.org/wp-content/uploads/2024-601-paper.pdf. [Last accessed on 6 Sep 2024]
34. Anada H. Decentralized multi-authority anonymous credential system with bundled languages on identifiers. In: Maimut D, Oprina AG, Sauveron D, editors. Innovative security solutions for information technology and communications. SecITC 2020. Lecture notes in computer science. Cham: Springer; 2020. pp. 71-90. DOI
35. Abe M, Fuchsbauer G, Groth J, Haralambiev K, Ohkubo M. Structure-preserving signatures and commitments to group elements. In: Rabin T, editor. Advances in Cryptology - CRYPTO 2010. Lecture notes in computer science. Berlin, Heidelberg: Springer; 2010. pp. 209-36. DOI
36. Goldwasser S, Micali S, Rivest RL. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J Comput* 1988;17:281-308. DOI
37. Wikipedia. Commitment scheme. Available from: https://en.wikipedia.org/wiki/Commitment_scheme. [Last accessed on 6 Sep 2024]
38. Abe M, Fuchsbauer G, Groth J, Haralambiev K, Ohkubo M. Structure-preserving signatures and commitments to group elements. *J Cryptol* 2016;29:363-421. DOI
39. TEPLA(University of Tsukuba Elliptic Curve and Pairing Library). (in Japanese) Available from: http://www.cipher.risk.tsukuba.ac.jp/tepla/doc/tepladoc2_0_0.pdf. [Last accessed on 6 Sep 2024]